Modicon Modbus Plus Host Interface Device Driver Manual for Windows 95

890 USE 125 00 Version 2.0

February 1998

GROUPE SCHNEIDER

■ Modicon ■ Square D ■ Telemecanique

Contents

Selecting Modbus Plus Libraries and Drivers

1

1.1 Modbus Plus Communications Overview

The Modbus Plus drivers described in this manual provide communications connectivity between software programs (such as Modsoft) and Schneider Automation's Modbus Plus PC network cards in a 32 bit Windows environment. These network cards are:

Adapter	**Part Number**
SA85 Single Channel Modbus Plus Adapter Card	AM-SA85-000
SA85 Dual Channel Modbus Plus Adapter Card	AM-SA85-002
PCMCIA Type II Modbus plus Adapter Card	416NHM21200
PCMCIA Type III Modbus Plus Adapter Card	416NHM21203

1.2 Application-Specific Drivers and Libraries

There are three components to the Modbus Plus MBX drivers. The selection of the correct driver components is determined by the requirements of your application.

CAUTION

MIXING DRIVERS WILL RESULT IN DRIVER FAILURE

Make sure you are using the correct driver components for your operating system. Windows 95 and Windows NT drivers are specific to Windows 95 and Windows NT operating systems, respectively. (For dual-boot system exception, see Chart Notes below. Mixing Windows 95 and Windows NT drivers within one operating system will result ina driver failure that may be difficult to correct.

Failure to observe this precaution can result in equipment damage.

The drivers are as follows:

- **The 32-bit MBX Driver** (previously called “Local NETLIB Library”) provides connectivity between Windows 95 or Windows NT based applications and Modbus Plus adapter cards. For Windows 95 use the SW-LNET-I95 driver. For Windows NT use SW-LNET-INT. Refer to the selection chart *Stand-alone or Remote Server Configurations* on page 3. See also *Example of a Stand-Alone Configuration* on page 5 for an illustration of how these libraries are applied.
- **The Virtual MBX Driver** (previously called “NETLIB/NetBIOS Virtual Device Driver” allows existing 16-bit DOS or Windows 3.1x applications to run under Windows 95 or Windows NT in their original binary form, without modification. For Windows 95 use the SW-WVDD-I95 Virtual Device Driver. For Windows NT use SW-WVDD-INT. Refer to the selection charts in *Modbus Plus Driver Selection Charts* on page 3. Also see *Example of a Stand-Alone Configuration* on page 5 for an illustration of the application of these drivers.
- **The 32-bit Remote MBX Driver** (previously called “Remote NETLIB Library”) provides connectivity between 32-bit Windows 95 or Windows NT based applications and Modbus Plus adapter cards running in remote Windows NT servers. For Windows 95 use the SW-RNET-I95 library. For Windows NT use SW-RNET-INT. Refer to the selection chart in *Remote Client Configurations* on page 4. Also see *Example of a Remote Client Configuration* on page 7 for an illustration of the application of these libraries.

1.3 Modbus Plus Driver Selection Charts

Use the following driver selection charts to determine the correct driver(s) for your operating system and application. Then refer to the appropriate chapter of this manual for detailed information.

Stand-alone or Remote Server Configurations

Use this chart to determine the correct driver(s) for your operating system and application in a Stand Alone (Local) configuration. The recommended order of installation is MBX (Local) Driver followed by Virtual Driver. See *Chart Notes* on page 4 for important information.

Adapter Type >>>	**SA85 Modbus Plus Adapter Card Single and Dual Channel**			**PCMCIA Modbus Plus Adapter Card Type II and Type III**		
Operating System	**16 Bit Applications**	**32 Bit Applications**	**Mixed 16 & 32 bit Applications**	**16 Bit Applications**	**32 Bit Applications**	**Mixed 16 & 32 bit Applications**
Win 95	SW-LNET-I95 and SW-WVDD-I95	SW-LNET-I95	SW-LNET-I95 and SW-WVDD-I95	SW-LNET-I95 and SW-WVDD-I95	SW-LNET-I95	SW-LNET-I95 and SW-WVDD-I95
Win NT	SW-LNET-INT and SW-WVDD-INT	SW-LNET-INT	SW-LNET-INT and SW-WVDD-INT	SW-LNET-INT and SW-WVDD-INT	SW-LNET-INT	SW-LNET-INT and SW-WVDD-INT

Remote Client Configurations

Use this chart to determine the correct driver(s) for your operating system and application when your Modbus Plus communications card is installed in a remote computer. The remote computer must be running SW-LNET-INT under Windows NT. The recommended order of installation is MBX (Local) Driver, then Virtual Driver if needed, followed by the Remote Driver. See *Chart Notes* on page 4 for important information.

Adapter Type >>>	**SA85 Modbus Plus Adapter Card Single and Dual Channel**			**PCMCIA Modbus Plus Adapter Card Type II and Type III**		
Operating System	**16 Bit Applications**	**32 Bit Applications**	**Mixed 16 & 32 bit Applications**	**16 Bit Applications**	**32 Bit Applications**	**Mixed 16 & 32 bit Applications**
Win 95	SW-RNET-I95 and SW-WVDD-I95	SW-RNET-I95	SW-RNET-I95 and SW-WVDD-I95	SW-RNET-I95 and SW-WVDD-I95	SW-RNET-I95	SW-RNET-I95 and SW-WVDD-I95
Win NT	SW-RNET-INT and SW-WVDD-INT	SW-RNET-INT	SW-RNET-INT and SW-WVDD-INT	SW-RNET-INT and SW-WVDD-INT	SW-RNET-INT	SW-RNET-INT and SW-WVDD-INT

Chart Notes

Dual-Boot Systems: Both Windows 95 and Windows NT MBX components may be installed on a dual-boot system, i.e., a system that optionally boots either Windows NT or Windows 95 optionally.

CAUTION

MIXING DRIVERS WILL RESULT IN DRIVER FAILURE

Make sure you are using the correct driver components for your operating system. Windows 95 and Windows NT drivers are specific to Windows 95 and Windows NT operating systems, respectively. (For dual-boot system exception, see Chart Notes below. Mixing Windows 95 and Windows NT drivers within one operating system will result ina driver failure that may be difficult to correct.

Failure to observe this precaution can result in equipment damage.

16 Bit Applications Include: ModLink, Modsoft, FactoryLink Version 4.x, ProWorx, and InTouch for Windows 95.

32 Bit Applications Include: FactoryLink ESC 6.xx and InTouch for Windows NT, plus any new applications that are based on Windows NT or Windows 95.

1.4 Example of a Stand-Alone Configuration

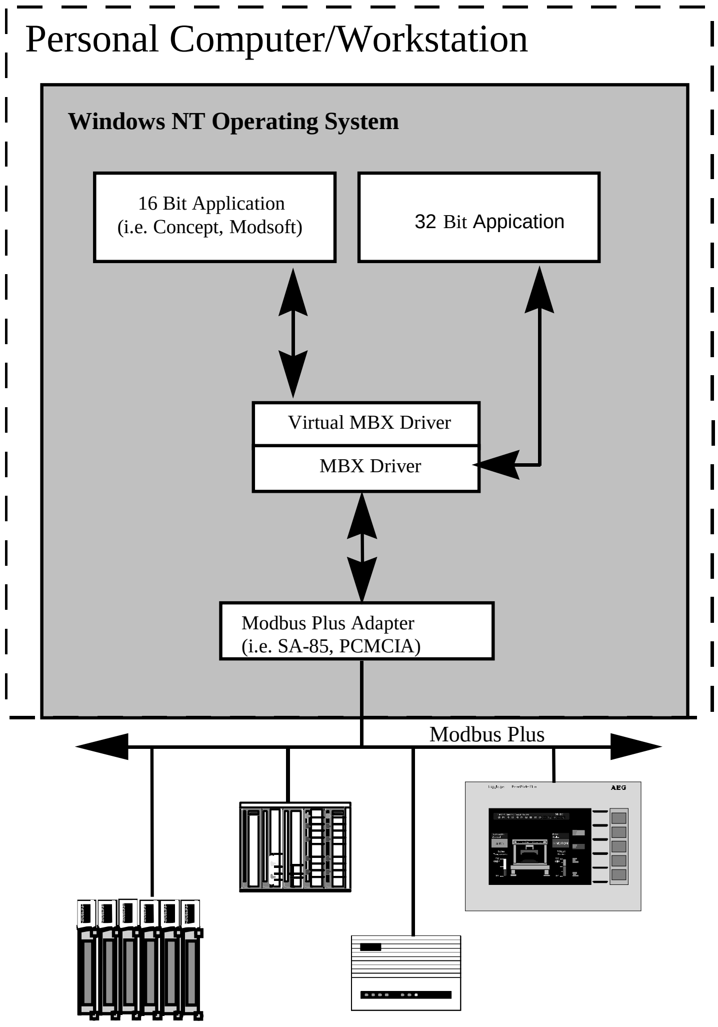
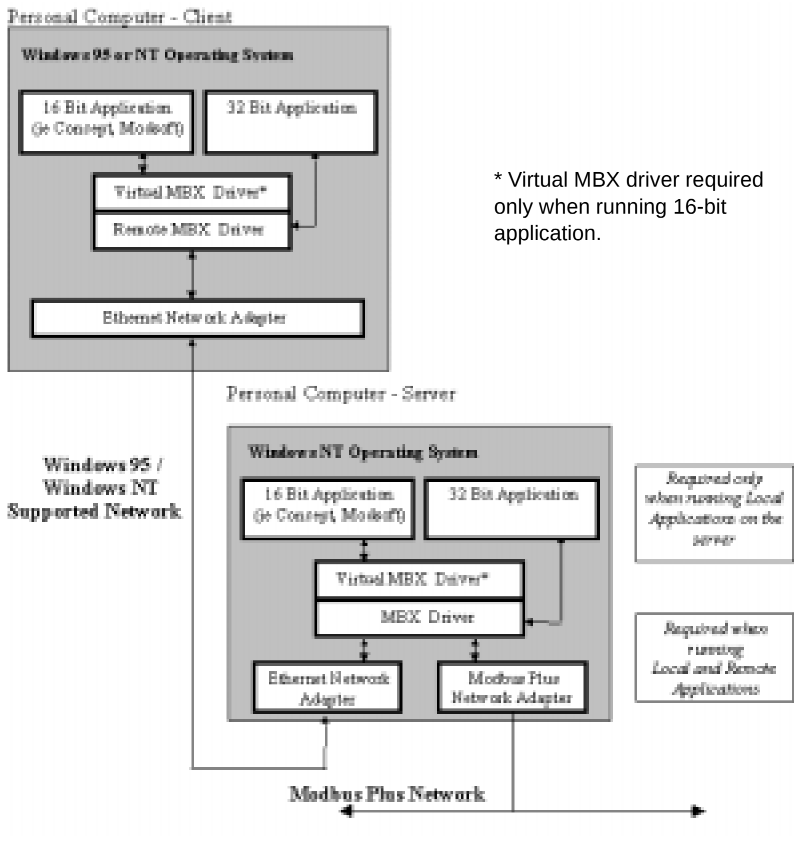


Figure 1 Example of a Stand-Alone Configuration

The following table lists the components of a stand-alone system:

Quantity	**Description**
1	Concept, Modsoft, Modlink, MBPSTAT, FactoryLink, other 16 or 32 bit application software
1	MBX (formerly called “Local”) Driver (SW-LNET-INT)
1	Virtual MBX Driver (SW-WVDD-INT)
1	Windows NT Operating System Software
1	Modbus Plus Host Interface Adapter (i.e., SA85, PCMCIA)

1.5 Example of a Remote Client Configuration


Personal Computer - Client
Windows 95 or NT Operating System
16 Bit Application (ie Concept, Modsoft)
32 Bit Application
Virtual MBX Driver*
Remote MBX Driver
Ethernet Network Adapter
Windows 95 / Windows NT Supported Network
Personal Computer - Server
Windows NT Operating System
16 Bit Application (ie Concept, Modsoft)
32 Bit Application
Virtual MBX Driver*
MBX Driver
Ethernet Network Adapter
Modbus Plus Network Adapter
Required only when running Local Applications on the server
Required when running Local and Remote Applications
Modbus Plus Network

* Virtual MBX driver required only when running 16-bit application.

Figure 2 Example of a Remote Client Configuration

The following table lists server components:

Quantity	Description
1	MBX Driver (SW-LNET-INT)
1	Virtual MBX Driver (SW-WVDD-INT) - The virtual MBX driver is only required when running 16-bit applications on the server.

The following table lists client components:

Quantity	Description
1	Remote MBX Driver (SW-RNET-INT or SW-RNET-I95)
1	Virtual MBX Driver (SW-WVDD-INT or SW-WVDD-I95) - The virtual MBX driver is only required when running 16-bit applications on the server.

MBX Driver

2

2.1 Introduction

The 32-bit MBX Driver for Microsoft Windows 95 operating system provides connectivity between Schneider Automation ModConnect host interface adapters and 32-bit applications running under Windows 95.

The kernel mode device driver that is part of this product is the highest performance Modbus Plus driver in the industry. The driver can operate in either *interrupt* or *polled mode* and supports all current Schneider Automation ModConnect host interface adapters for ISA, EISA, MCA, and PC Card (PCMCIA) buses. Multiple interface cards can be installed at the same time, limited only by the number of available slots. Full implementation of all MB+ features provides support for *Data Master/Slave*, *Program Master/Slave*, and *Global Data*. The high performance native API of the MBX Driver is designed to take advantage of all features of the event-driven, multitasking, multithreaded Windows 95 operating system.

The driver provides compatibility with applications written to utilize its high performance application programming interface (API) as well as the industry standard NETLIB interface specification from Schneider Automation. The 32-bit NETLIB compatibility provides an excellent bridge for developers who would like to port their 16-bit, NETLIB-compatible applications to Windows 95. Developers of new applications can use either the NETLIB or the high performance MBXAPI programming interface. To obtain the MBXAPI software development kit, including the MBXAPI specification, MBXAPI sample source code, and NETLIB sample source code, contact Schneider Automation. For complete reference of all NETLIB library functions refer to "*Modicon IBM Host Based Devices User's Guide*" from Schneider Automation (890 USE 102 00).

Running 16-Bit Software

A companion product, the Virtual MBX Driver, will allow all 16-bit NETLIB/NetBIOS-compatible applications (e.g., ModSoft) to run concurrently with all 32-bit applications in the same computer. The Virtual MBX Driver will allow multiple 16-bit applications as well as multiple instances of a single 16-bit application to run under the 32-bit Windows 95 operating system. (The Virtual MBX Driver is purchased separately and it requires either the MBX Driver or the Remote MBX Driver to operate).

2.2 Installation

Procedure

Windows 95 Installation Procedure

1. Exit all applications that are using MBX products like the *MBX Driver*, the *Remote MBX Driver*, or the *Virtual MBX Driver* before installing the new version. If the Virtual MBX Driver is running, stop it before beginning the installation.
2. Place the installation disk in the floppy drive (either Drive A: or B:).
3. Go to the *Control Panel*, select the ***Add/Remove Programs*** icon, and click the ***Install*** button. click ***Next>*** and then ***Finish*** to install the software onto the hard drive.
4. Configure the MBX Driver to match the installed adapter card(s) See *MBX Driver Configuration* on page 11 for more information on configuring adapters.

2.3 MBX Driver Configuration

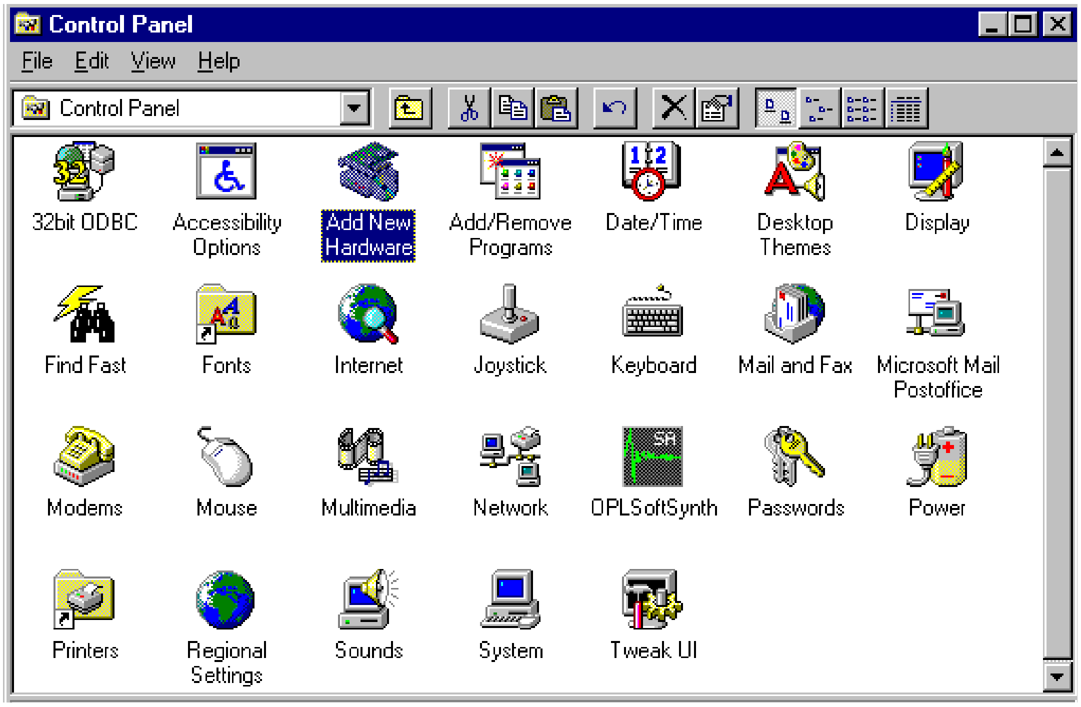
The MBX Driver includes a kernel mode device driver that requires a number of parameters to be specifically set to match the configuration of the devices used in a system. The following sections describe the process for adding new adapters and modifying the configurations of existing adapters.

During startup the driver may detect a number of configuration problems. When a problem is detected, the driver logs error messages. Refer to *Troubleshooting the MBX Driver* on page 42 for more information.

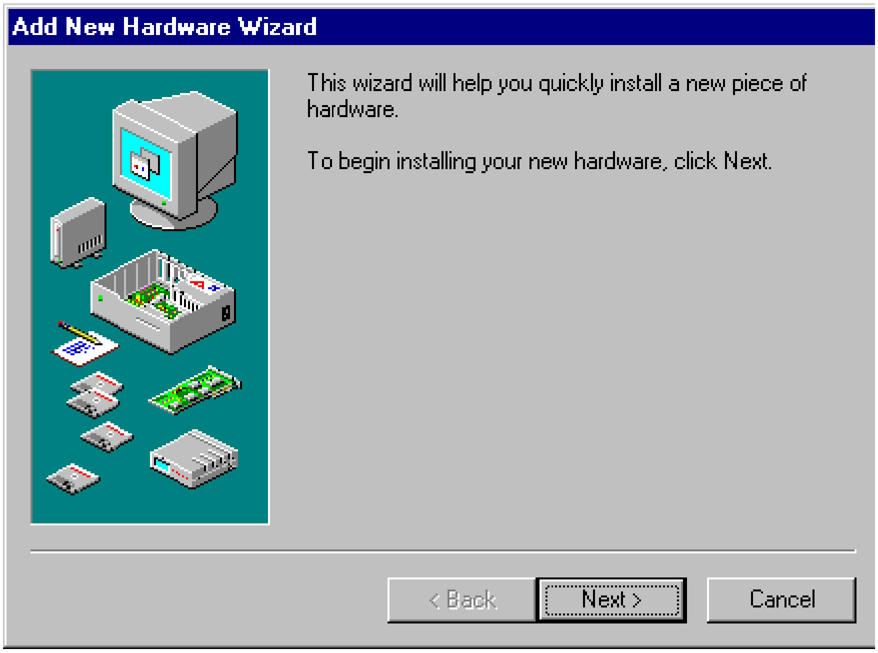
2.3.1 First Time Configuration

Procedure **First Time Configuration**

1. Go to the ***Control Panel*** and double click on the ***Add New Hardware*** icon.



2. Click ***Next >*** when the following dialog appears.

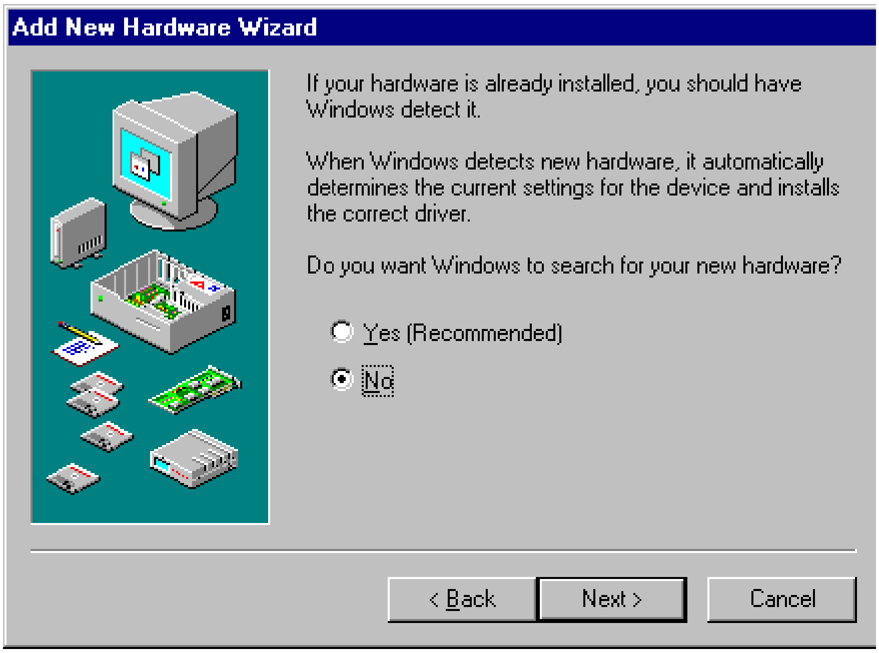


Add New Hardware Wizard

This wizard will help you quickly install a new piece of hardware.

To begin installing your new hardware, click Next.

3. When asked if Windows should search for the new hardware, select ***No*** and click ***Next >***.



Add New Hardware Wizard

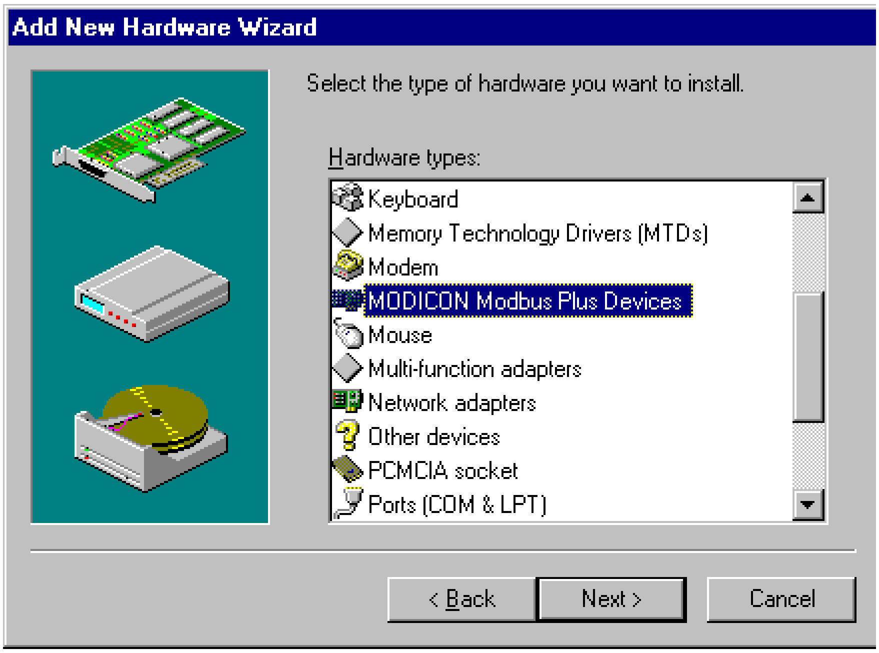
If your hardware is already installed, you should have Windows detect it.

When Windows detects new hardware, it automatically determines the current settings for the device and installs the correct driver.

Do you want Windows to search for your new hardware?

Yes (Recommended) [✓ No]

4. For the hardware type, select ***MODICON Modbus Plus Devices***.

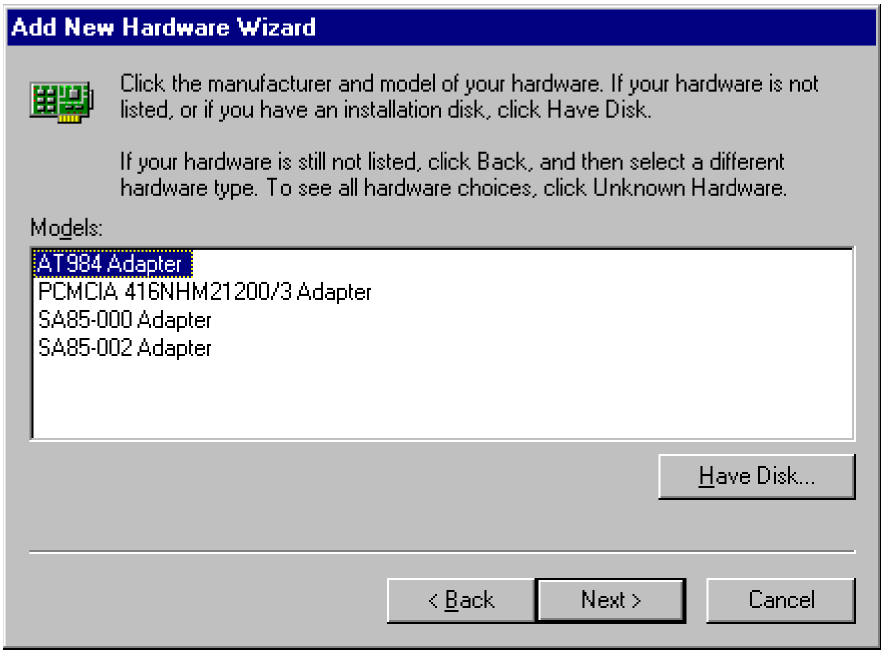

Add New Hardware Wizard

Select the type of hardware you want to install.

Hardware types:

- Keyboard
- Memory Technology Drivers (MTDs)
- Modem
- MODICON Modbus Plus Devices
- Mouse
- Multi-function adapters
- Network adapters
- Other devices
- PCMCIA socket
- Ports (COM & LPT)

5. Select the device type from the list that matches the hardware that is installed in the system. Click ***Next >***.



Add New Hardware Wizard

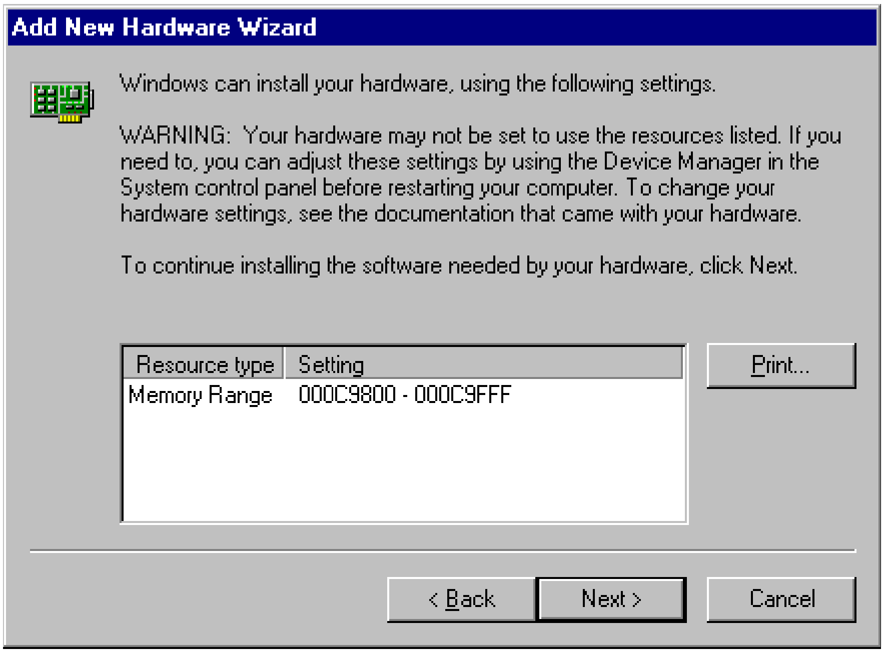
Click the manufacturer and model of your hardware. If your hardware is not listed, or if you have an installation disk, click Have Disk.

If your hardware is still not listed, click Back, and then select a different hardware type. To see all hardware choices, click Unknown Hardware.

Models:

AT984 Adapter
PCMCIA 416NHM21200/3 Adapter
SA85-000 Adapter
SA85-002 Adapter

6. At this point, a memory address is automatically chosen. (Note that the memory address selected by your system may not match the address shown in the following figure.) Click ***Next>***.



Add New Hardware Wizard

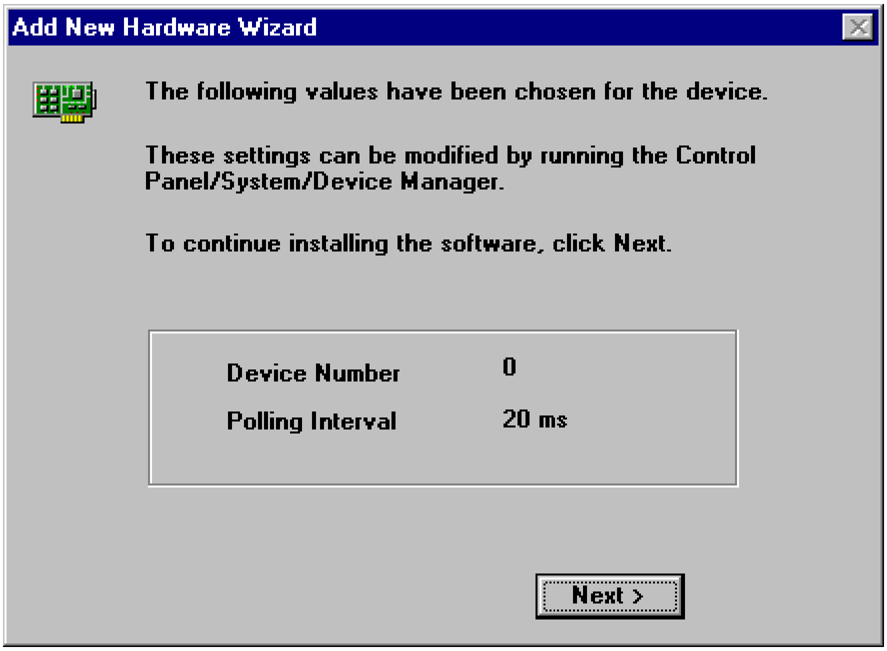
Windows can install your hardware, using the following settings.

WARNING: Your hardware may not be set to use the resources listed. If you need to, you can adjust these settings by using the Device Manager in the System control panel before restarting your computer. To change your hardware settings, see the documentation that came with your hardware.

To continue installing the software needed by your hardware, click Next.

Resource type	Setting
Memory Range	000C9800 - 000C9FFF

7. Polled mode (with a polling rate of 20 milliseconds) is the default mode of operation. (Note the device number selected by your system may not match the values shown in the following figure. See *Editing Current Configuration* on page 19 for information on changing the device settings.) Click ***Next>***.



Add New Hardware Wizard

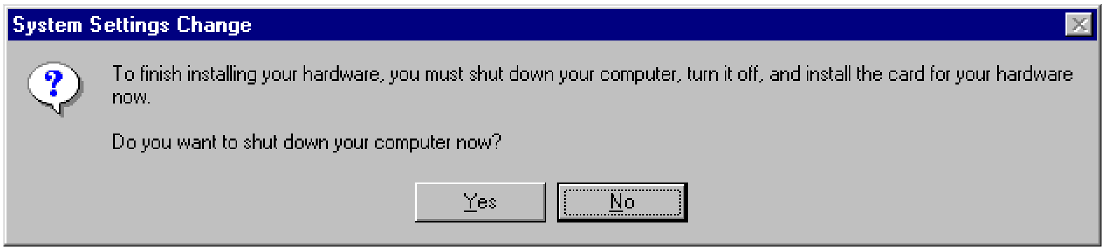
The following values have been chosen for the device.

These settings can be modified by running the Control Panel/System/Device Manager.

To continue installing the software, click Next.

Device Number	0
Polling Interval	20 ms

8. The device configuration will be modified first. After that is done, the system will again give you the option of shutting down and modifying the hardware settings. Select **No**. See *Editing Current Configuration* on page 19 for information on changing the device settings.

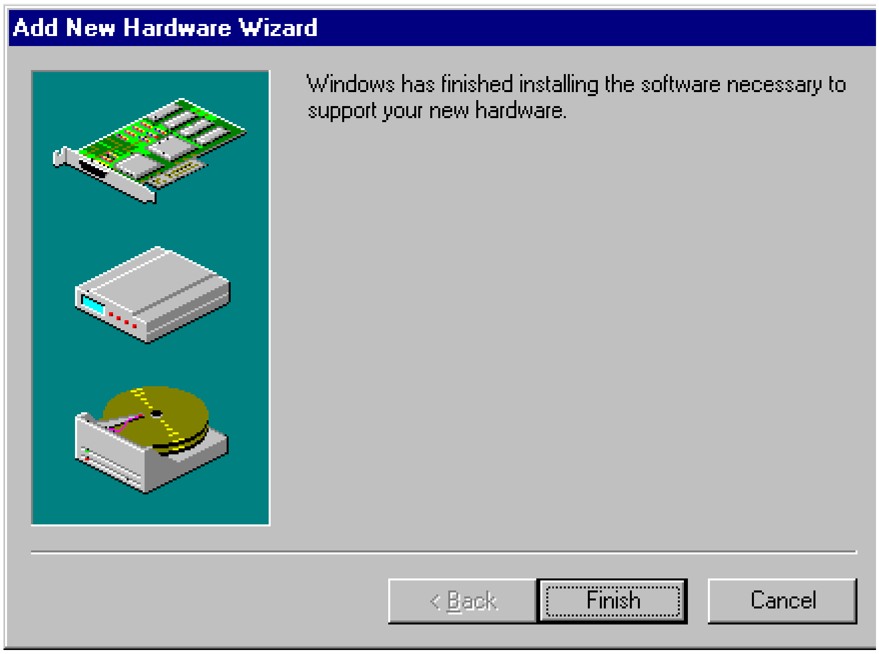


System Settings Change

To finish installing your hardware, you must shut down your computer, turn it off, and install the card for your hardware now.

Do you want to shut down your computer now?

9. Select ***Finish***.



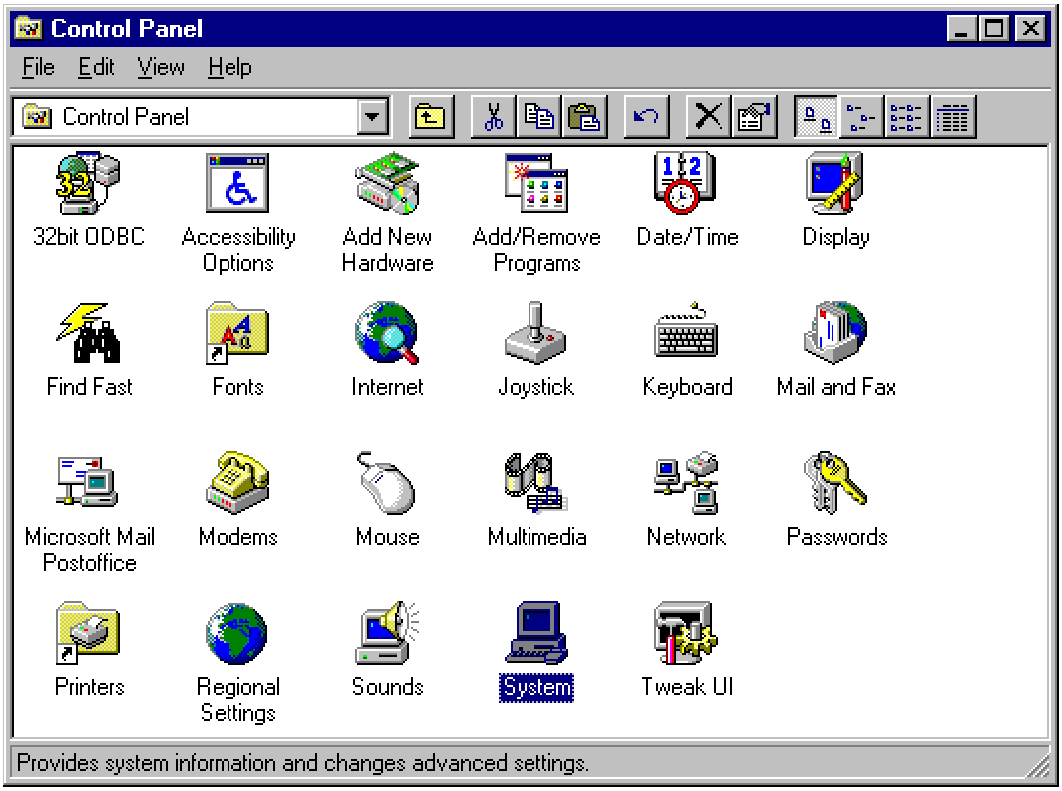
Add New Hardware Wizard

Windows has finished installing the software necessary to support your new hardware.

2.3.2 Editing Current Configuration

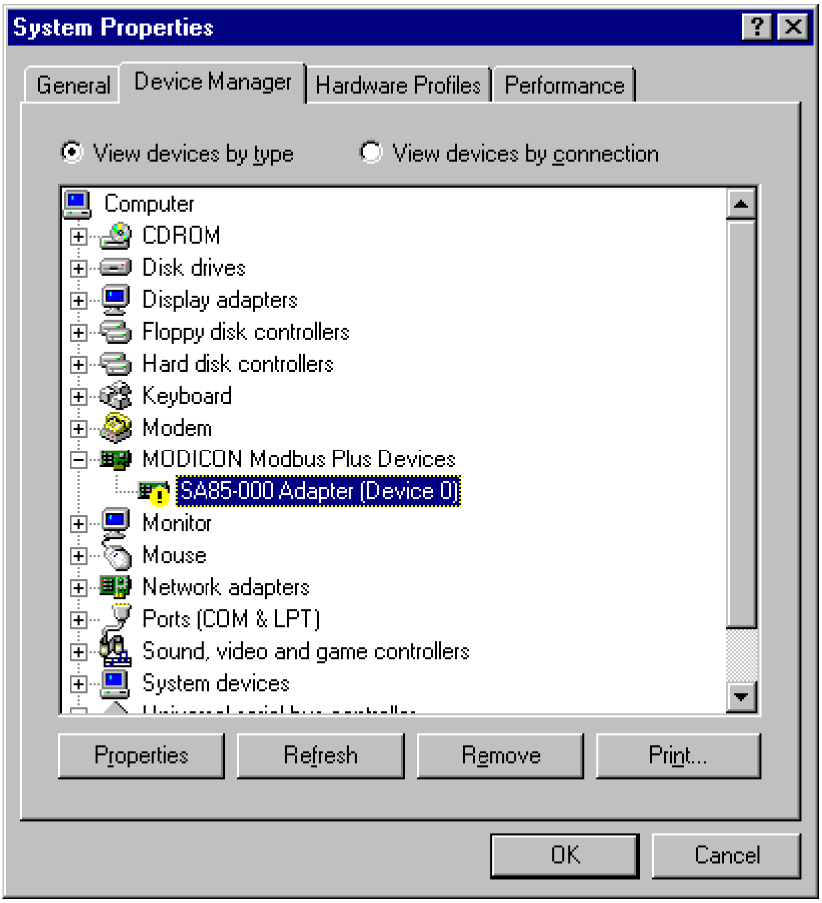
Procedure **Editing Current Configuration**

1. Go to the ***Control Panel*** and double click on the ***System*** icon.



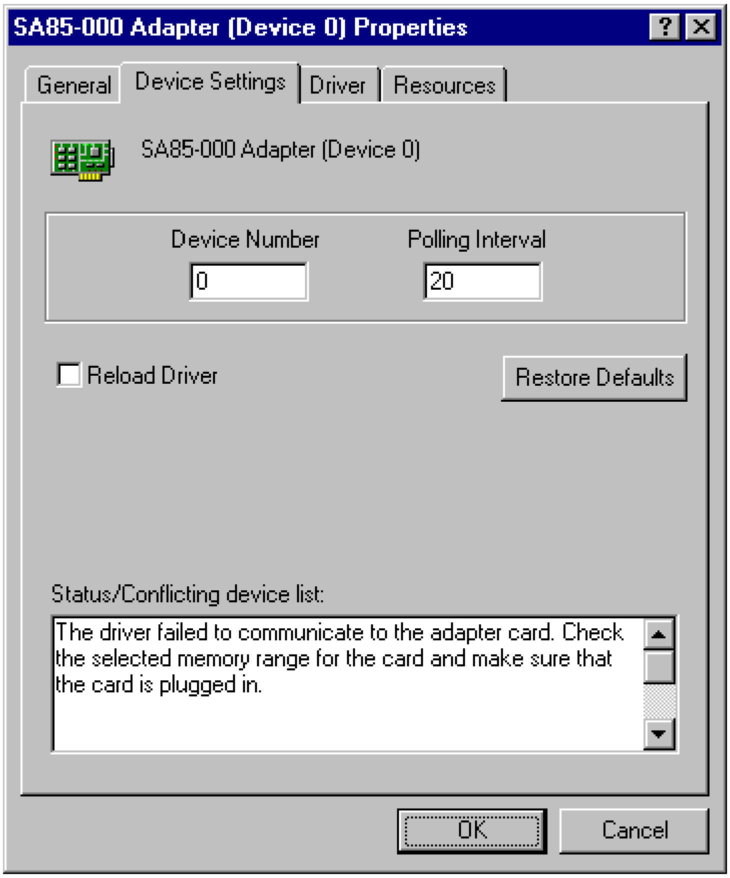
Provides system information and changes advanced settings.

2. Choose the ***Device Manager*** tab. Drill down into the ***MODICON Modbus Plus Devices*** item. Select the device that needs its configuration modified and click the ***Properties*** button.



3. At this point, a property sheet is displayed. See the following sections for specific configuration information about each device type.

2.3.3 SA85-000 Adapter



Device Number: 0

Polling Interval: 20

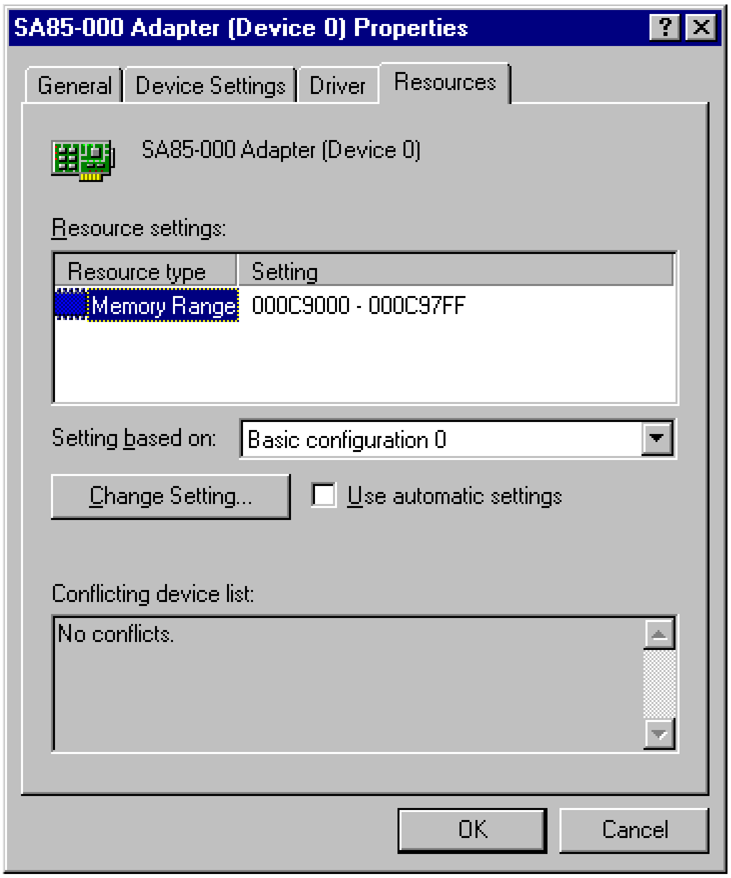
☐ Reload Driver

Status/Conflicting device list:

The driver failed to communicate to the adapter card. Check the selected memory range for the card and make sure that the card is plugged in.

- **Device Number**: A number that you assign to every adapter card installed in your system. **This is not the Modbus Plus node address!** By default, the system will try to use consecutive device numbers for the devices, starting from zero. However, this is not a requirement, and the user can assign numbers that are not consecutive. However, when running Modsoft or Concept, adapter numbers 0 or 1 must be used.
- **Polling Interval**: This parameter specifies the polling interval in milliseconds that the driver will use when running in the polled mode. The valid range for the polling interval is 20-1000 milliseconds. The default value is 20 ms.
- **Reload Driver**: When this box is checked, the driver will reload using the new configuration parameters after the OK button is clicked.

- **Status/Conflicting Device List**: Contains any error messages from the driver, along with a list of devices that conflict with the current device. See *Troubleshooting the MBX Driver* on page 42 for a list of possible error messages.



SA85-000 Adapter (Device 0) Properties

SA85-000 Adapter (Device 0)

Resource settings:

Resource type	Setting
Memory Range	000C9000 - 000C97FF

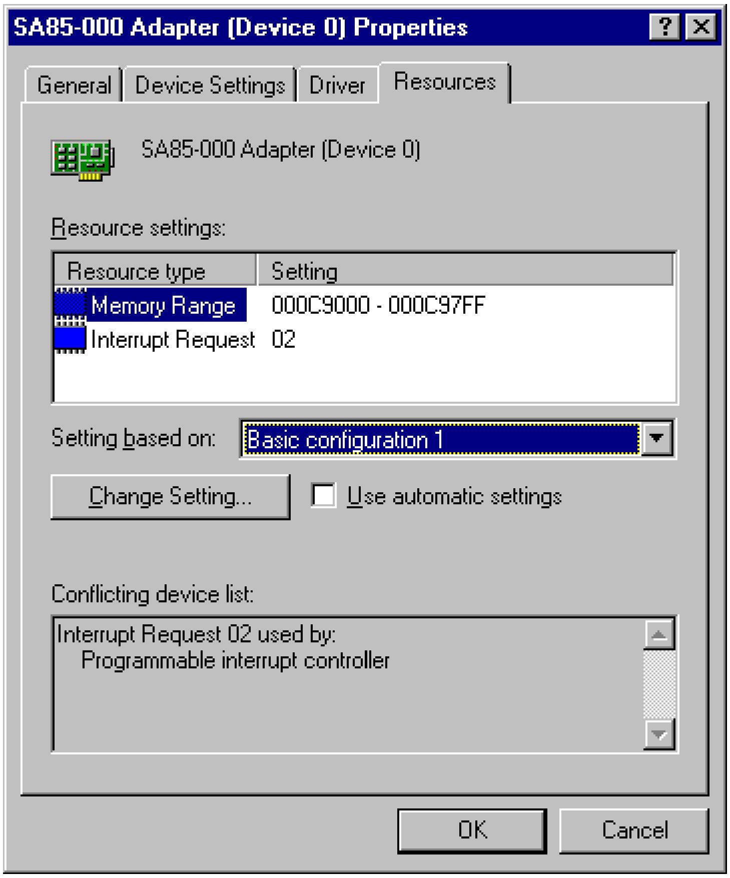
Setting based on: Basic configuration 0

Conflicting device list:

No conflicts.

- **Memory Range**: This parameter specifies the base address of the adapter card (e.g. D0000). This address must match the switch settings on the card and has to be unique for each adapter card.

- **Polled mode/Interrupt driven**: The kernel mode device driver can operate in either polled mode or interrupt mode. Select the proper mode of operation depending on the adapter card configuration. (The interrupt mode provides better performance than the polled mode. However, interrupt mode requires more processor overhead). It is legal to mix interrupt and polled modes of operation for different cards in the same system. The default for this parameter is polled mode. Basic configuration 0 is polled mode. Basic configuration 1 is interrupt mode.



SA85-000 Adapter (Device 0)

Resource settings:

Resource type	Setting
Memory Range	000C9000 - 000C97FF
Interrupt Request	02

Setting based on: Basic configuration 1

Change Setting...

Use automatic settings

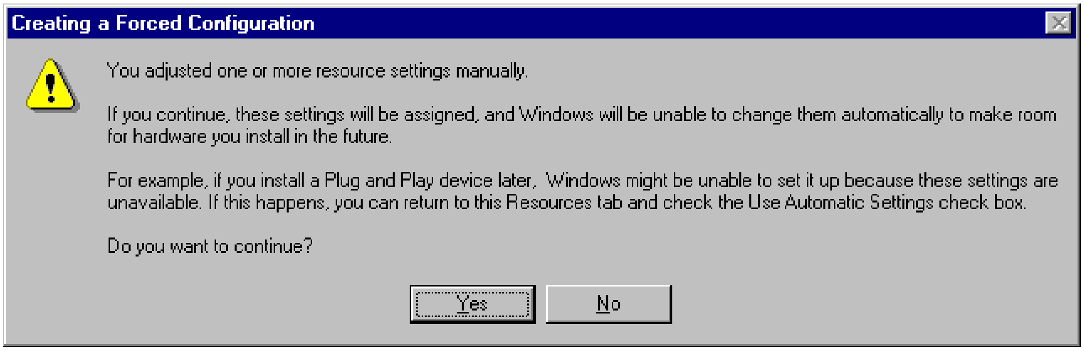
Conflicting device list:

Interrupt Request 02 used by:
Programmable interrupt controller

- **Memory Range**: This parameter specifies the base address of the adapter card (e.g. D0000). This address must match the switch settings on the card and has to be unique for each adapter card.

- **Polled mode/Interrupt driven**: The kernel mode device driver can operate in either polled mode or interrupt mode. Select the proper mode of operation depending on the adapter card configuration. (The interrupt mode provides better performance than the polled mode. However, interrupt mode requires more processor overhead). It is legal to mix interrupt and polled modes of operation for different cards in the same system. The default for this parameter is polled mode. Basic configuration 0 is polled mode. Basic configuration 1 is interrupt mode.
- **Interrupt Request**: When interrupt mode is selected, this parameter specifies the IRQ number for the interrupt line used (e.g. 5). This IRQ number must match the IRQ setting on the adapter card and has to be a unique value for each card in the system.

Once ***OK*** is clicked, the following *Creating a Forced Configuration* dialog may be displayed. If it is, select ***Yes***.



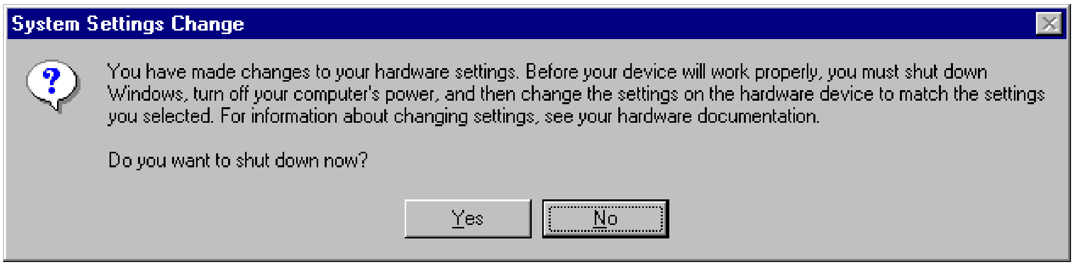
You adjusted one or more resource settings manually.

If you continue, these settings will be assigned, and Windows will be unable to change them automatically to make room for hardware you install in the future.

For example, if you install a Plug and Play device later, Windows might be unable to set it up because these settings are unavailable. If this happens, you can return to this Resources tab and check the Use Automatic Settings check box.

Do you want to continue?

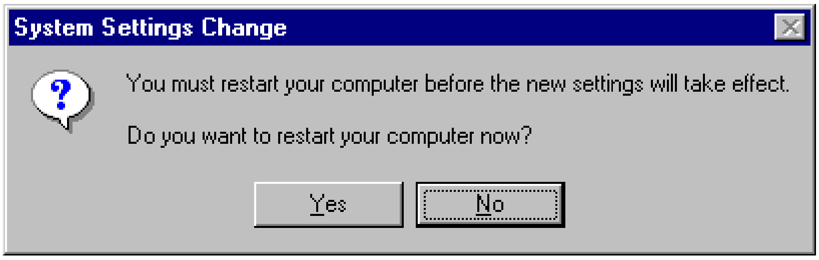
Next, the *System Settings Change* dialog may be displayed. If the physical settings on the adapter card do <u>NOT</u> need to be changed, select ***No***. If the adapter card's settings do need to be changed, select ***Yes***.



You have made changes to your hardware settings. Before your device will work properly, you must shut down Windows, turn off your computer's power, and then change the settings on the hardware device to match the settings you selected. For information about changing settings, see your hardware documentation.

Do you want to shut down now?

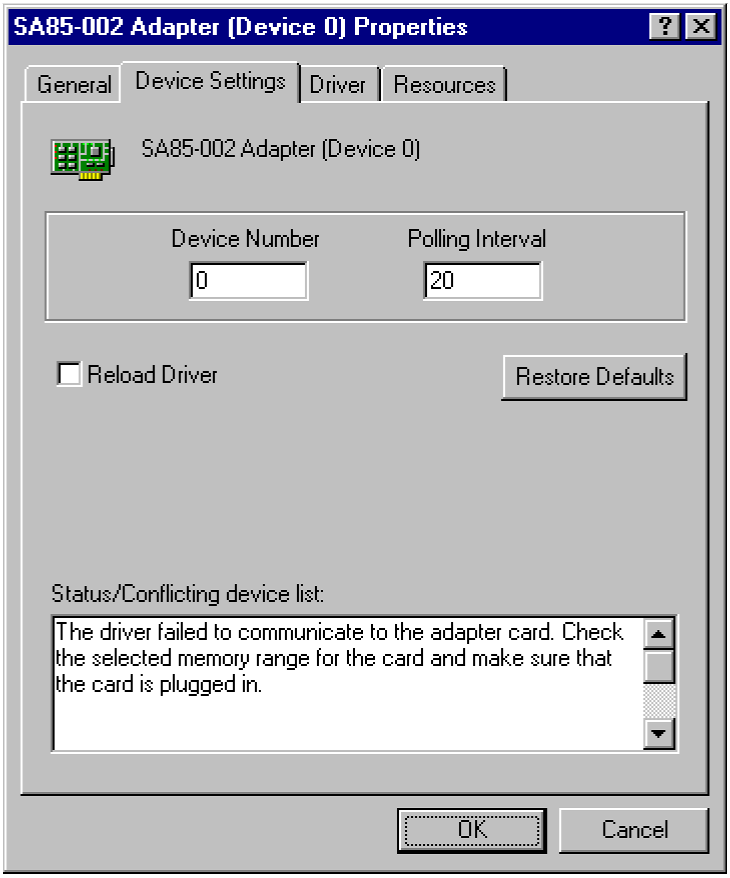
Finally, the *System Settings Change* dialog may be displayed. Since the MBX Driver is Plug and Play, the system can restart the driver without restarting the system. Select either ***Yes*** or ***No***.



You must restart your computer before the new settings will take effect.

Do you want to restart your computer now?

2.3.4 SA85-002 Adapter


SA85-002 Adapter (Device 0) Properties

SA85-002 Adapter (Device 0)

Device Number: 0

Polling Interval: 20

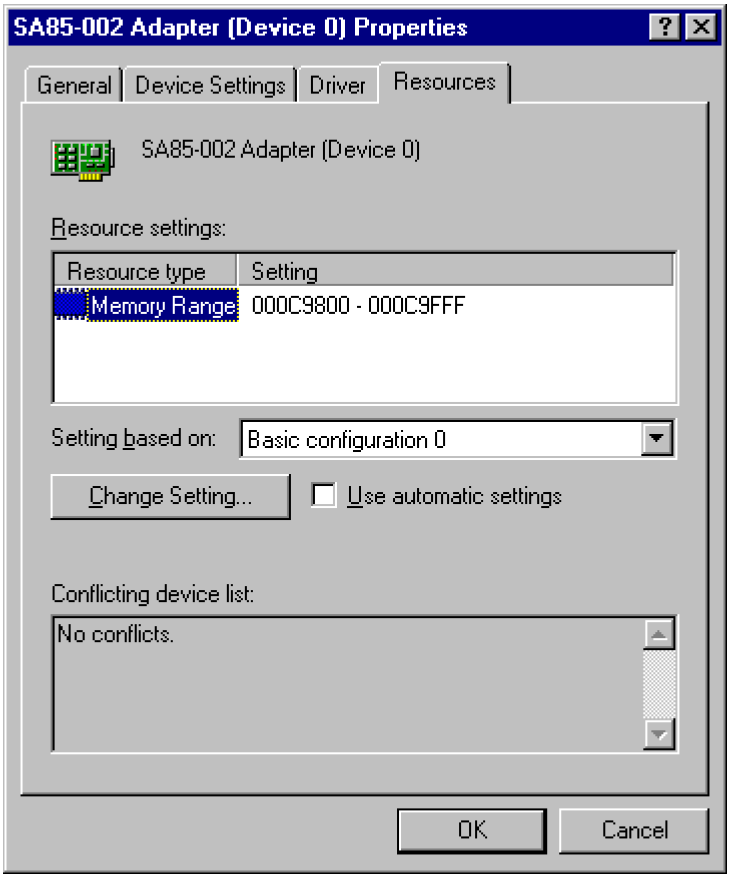
Reload Driver

Status/Conflicting device list:

The driver failed to communicate to the adapter card. Check the selected memory range for the card and make sure that the card is plugged in.

- **Device Number**: A number that you assign to every adapter card installed in your system. **This is not the Modbus Plus node address!** By default, the system will try to use consecutive device numbers for the devices, starting from zero. This is not a requirement, and the user can assign numbers that are not consecutive. However, when running Modsoft or Concept, adapter numbers 0 or 1 must be used.
- **Polling Interval**: This parameter specifies the polling interval in milliseconds that the driver will use when running in the polled mode. The valid range for the polling interval is 20-1000 milliseconds. The default value is 20 ms.
- **Reload Driver**: When this box is checked, the driver will reload using the new configuration parameters after the OK button is clicked.

- **Status/Conflicting Device List**: Contains any error messages from the driver, along with a list of devices that conflict with the current device. See *Troubleshooting the MBX Driver* on page 42 for a list of possible error messages.



SA85-002 Adapter (Device 0)

Resource settings:

Resource type	Setting
Memory Range	000C9800 - 000C9FFF

Setting based on: Basic configuration 0

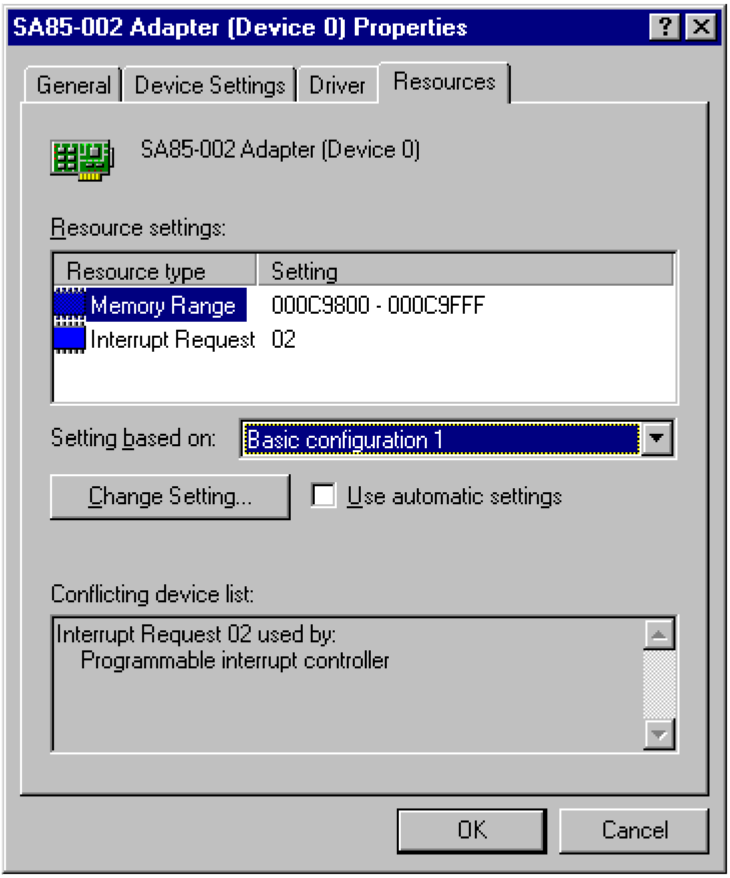
☐ Use automatic settings

Conflicting device list:

No conflicts.

- **Memory Range**: This parameter specifies the base address of the adapter card (e.g. D0000). This address must match the switch settings on the card and has to be unique for each adapter card.

- **Polled mode/Interrupt driven**: The kernel mode device driver can operate in either polled mode or interrupt mode. Select the proper mode of operation depending on the adapter card configuration. (The interrupt mode provides better performance than the polled mode. However, interrupt mode requires more processor overhead). It is legal to mix interrupt and polled modes of operation for different cards in the same system. The default for this parameter is polled mode. Basic configuration 0 is polled mode. Basic configuration 1 is interrupt mode.



SA85-002 Adapter (Device 0) Properties

SA85-002 Adapter (Device 0)

Resource settings:

Resource type	Setting
Memory Range	000C9800 - 000C9FFF
Interrupt Request	02

Setting based on: Basic configuration 1

☐ Use automatic settings

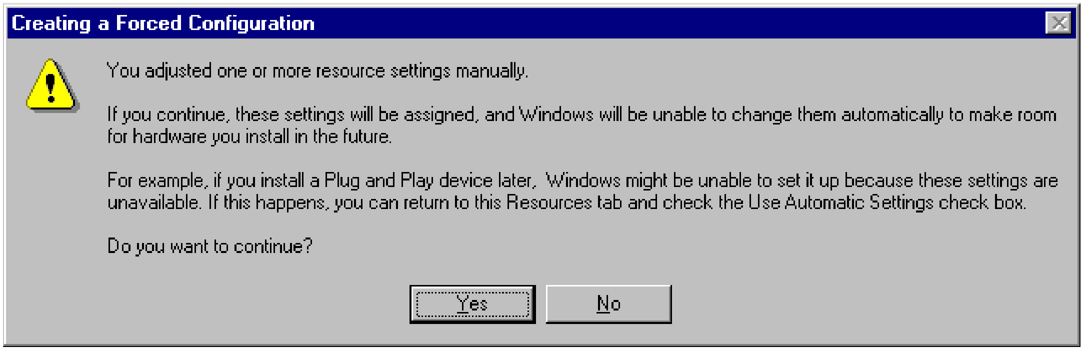
Conflicting device list:

Interrupt Request 02 used by:
Programmable interrupt controller

- **Memory Range**: This parameter specifies the base address of the adapter card (e.g. D0000). This address must match the switch settings on the card and has to be unique for each adapter card.

- **Polled mode/Interrupt driven**: The kernel mode device driver can operate in either polled mode or interrupt mode. Select the proper mode of operation depending on the adapter card configuration. (The interrupt mode provides better performance than the polled mode. However, interrupt mode requires more processor overhead). It is legal to mix interrupt and polled modes of operation for different cards in the same system. The default for this parameter is polled mode. Basic configuration 0 is polled mode. Basic configuration 1 is interrupt mode.
- **Interrupt Request**: When interrupt mode is selected, this parameter specifies the IRQ number for the interrupt line used (e.g. 5). This IRQ number must match the IRQ setting on the adapter card and has to be a unique value for each card in the system.

Once ***OK*** is clicked, the following *Creating a Forced Configuration* dialog may be displayed. If it is, select ***Yes***.


Creating a Forced Configuration

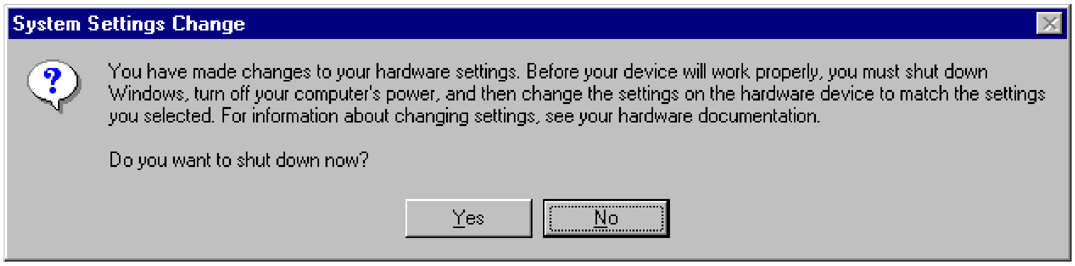
You adjusted one or more resource settings manually.

If you continue, these settings will be assigned, and Windows will be unable to change them automatically to make room for hardware you install in the future.

For example, if you install a Plug and Play device later, Windows might be unable to set it up because these settings are unavailable. If this happens, you can return to this Resources tab and check the Use Automatic Settings check box.

Do you want to continue?

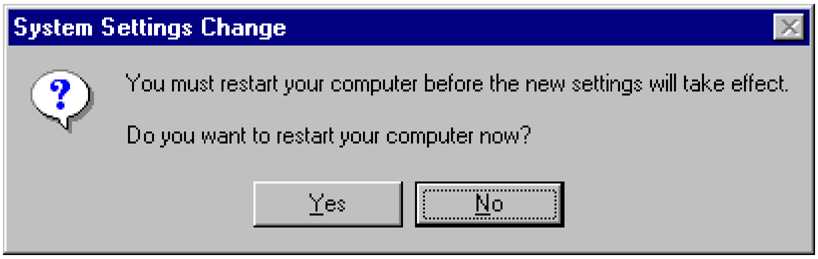
Next, the *System Settings Change* dialog may be displayed. If the physical settings on the adapter card do NOT need to be changed, select ***No***. If the adapter card's settings do need to be changed, select ***Yes***.


System Settings Change

You have made changes to your hardware settings. Before your device will work properly, you must shut down Windows, turn off your computer's power, and then change the settings on the hardware device to match the settings you selected. For information about changing settings, see your hardware documentation.

Do you want to shut down now?

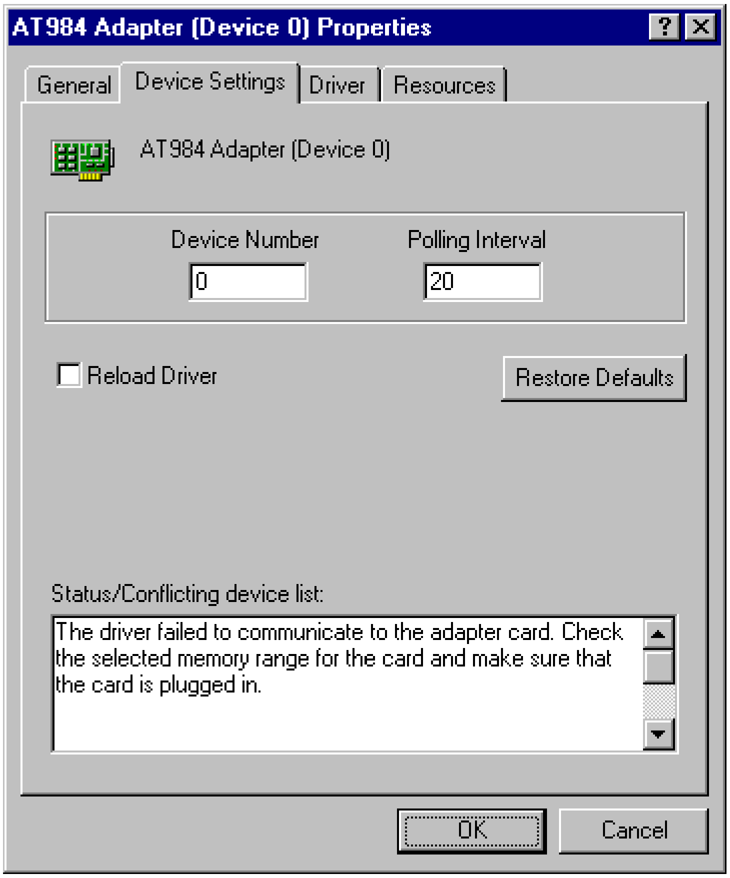
Finally, the *System Settings Change* dialog may be displayed. Since the MBX Driver is Plug and Play, the system can restart the driver without restarting the system. Select either ***Yes*** or ***No***.


System Settings Change

You must restart your computer before the new settings will take effect.

Do you want to restart your computer now?

2.3.5 AT984 Adapter



Device Number: 0

Polling Interval: 20

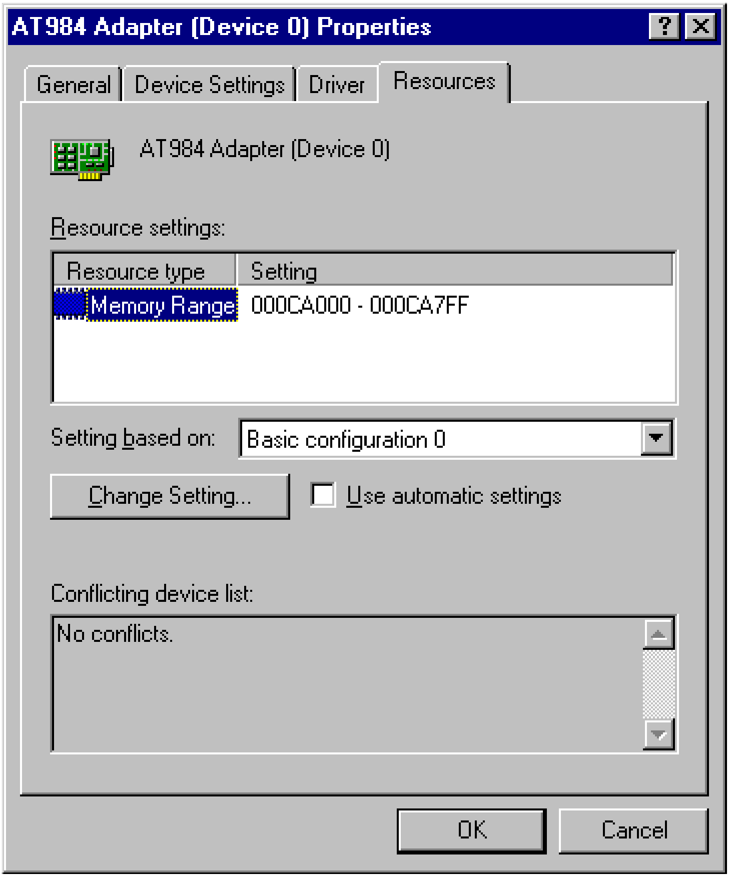
☐ Reload Driver

Status/Conflicting device list:

The driver failed to communicate to the adapter card. Check the selected memory range for the card and make sure that the card is plugged in.

- **Device Number**: A number that you assign to every device installed in your system. **This is not the Modbus Plus node address!** By default, the system will try to use consecutive device numbers for the devices, starting from zero. However, this is not a requirement, and the user can assign numbers that are not consecutive. However, when running Modsoft or Concept, adapter numbers 0 or 1 must be used.
- **Polling Interval**: This parameter specifies the polling interval in milliseconds that the driver will use when running in the polled mode. The valid range for the polling interval is 20-1000 milliseconds. The default value is 20 ms.
- **Reload Driver**: When this box is checked, the driver will reload using the new configuration parameters after the OK button is clicked.

- **Status/Conflicting Device List**: Contains any error messages from the driver, along with a list of devices that conflict with the current device. See *Troubleshooting the MBX Driver* on page 42 for a list of possible error messages.



AT984 Adapter (Device 0)

Resource settings:

Resource type	Setting
Memory Range	000CA000 - 000CA7FF

Setting based on: Basic configuration 0

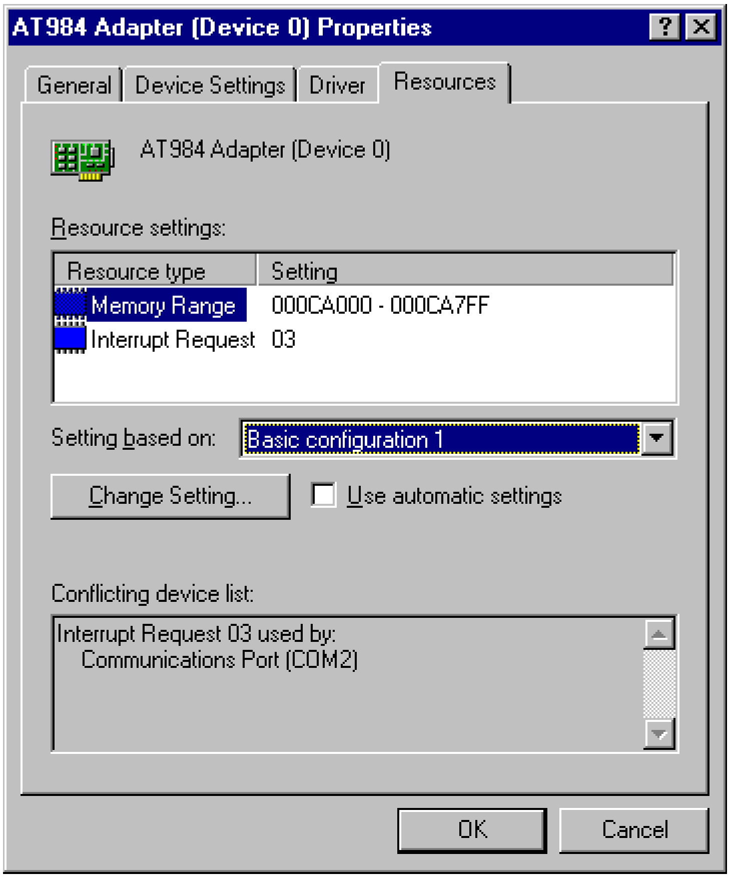
☐ Use automatic settings

Conflicting device list:

No conflicts.

- **Memory Range**: This parameter specifies the base address of the adapter card (e.g. D0000). This address must match the switch settings on the card and has to be unique for each adapter card.

- **Polled mode/Interrupt driven**: The kernel mode device driver can operate in either polled mode or interrupt mode. Select the proper mode of operation depending on the adapter card configuration. (The interrupt mode provides better performance than the polled mode. However, interrupt mode requires more processor overhead). It is legal to mix interrupt and polled modes of operation for different cards in the same system. The default for this parameter is polled mode. Basic configuration 0 is polled mode. Basic configuration 1 is interrupt mode.


AT984 Adapter (Device 0) Properties

AT984 Adapter (Device 0)

Resource settings:

Resource type	Setting
Memory Range	000CA000 - 000CA7FF
Interrupt Request	03

Setting based on: Basic configuration 1

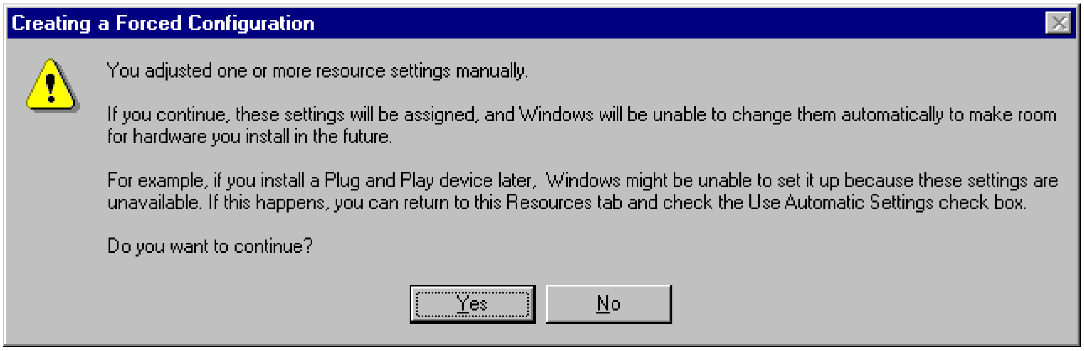
Conflicting device list:

Interrupt Request 03 used by:
Communications Port (COM2)

- **Memory Range**: This parameter specifies the base address of the adapter card (e.g. D0000). This address must match the switch settings on the card and has to be unique for each adapter card.

- **Polled mode/Interrupt driven**: The kernel mode device driver can operate in either polled mode or interrupt mode. Select the proper mode of operation depending on the adapter card configuration. (The interrupt mode provides better performance than the polled mode. However, interrupt mode requires more processor overhead). It is legal to mix interrupt and polled modes of operation for different cards in the same system. The default for this parameter is polled mode. Basic configuration 0 is polled mode. Basic configuration 1 is interrupt mode.
- **Interrupt Request**: When interrupt mode is selected, this parameter specifies the IRQ number for the interrupt line used (e.g. 5). This IRQ number must match the IRQ setting on the adapter card and has to be a unique value for each card in the system.

Once ***OK*** is clicked, the following *Creating a Forced Configuration* dialog may be displayed. If it is, select ***Yes***.


Creating a Forced Configuration

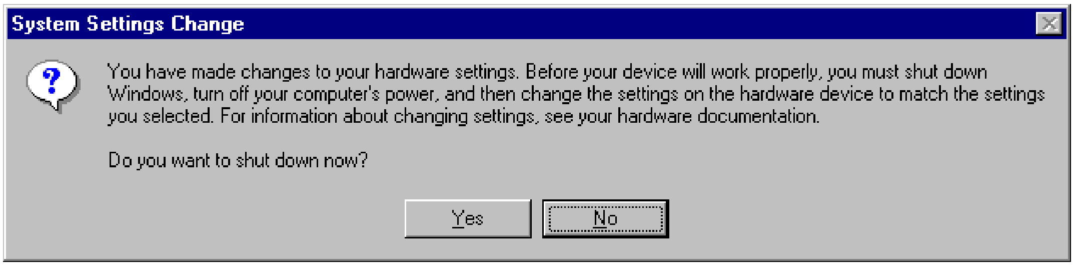
You adjusted one or more resource settings manually.

If you continue, these settings will be assigned, and Windows will be unable to change them automatically to make room for hardware you install in the future.

For example, if you install a Plug and Play device later, Windows might be unable to set it up because these settings are unavailable. If this happens, you can return to this Resources tab and check the Use Automatic Settings check box.

Do you want to continue?

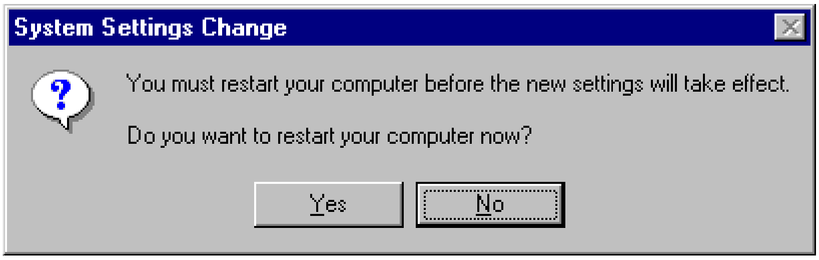
Next, the *System Settings Change* dialog may be displayed. If the physical settings on the adapter card do NOT need to be changed, select ***No***. If the adapter card's settings do need to be changed, select ***Yes***.


System Settings Change

You have made changes to your hardware settings. Before your device will work properly, you must shut down Windows, turn off your computer's power, and then change the settings on the hardware device to match the settings you selected. For information about changing settings, see your hardware documentation.

Do you want to shut down now?

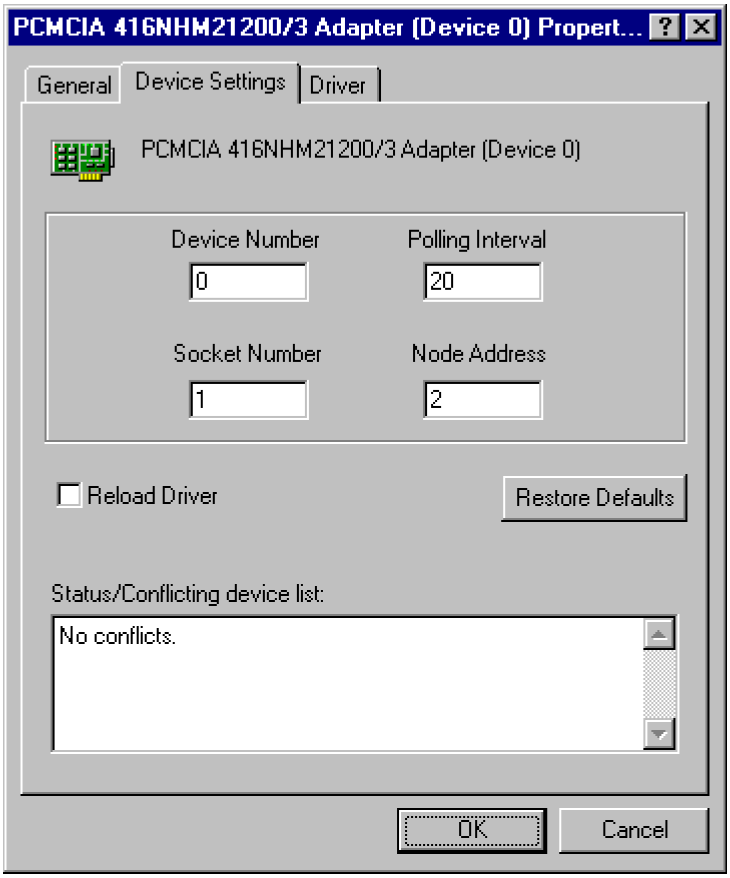
Finally, the *System Settings Change* dialog may be displayed. Since the MBX Driver is Plug and Play, the system can restart the driver without restarting the system. Select either ***Yes*** or ***No***.


System Settings Change

You must restart your computer before the new settings will take effect.

Do you want to restart your computer now?

2.3.6 PCMCIA 416NHM21200/3 Adapter


PCMCIA 416NHM21200/3 Adapter (Device 0) Propert...

PCMCIA 416NHM21200/3 Adapter (Device 0)

Device Number: 0
Polling Interval: 20
Socket Number: 1
Node Address: 2

☐ Reload Driver

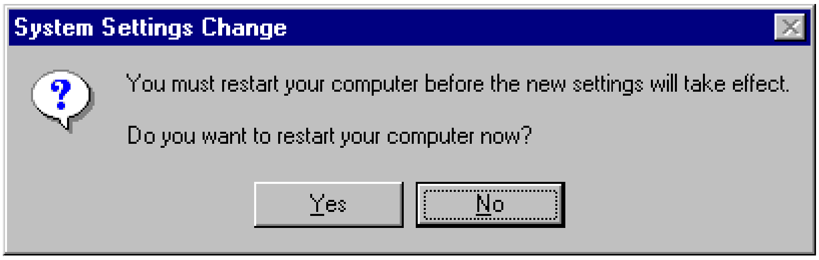
Status/Conflicting device list:

No conflicts.

- **Device Number**: A number that you assign to every device installed in your system. **This is not the Modbus Plus node address!** By default, the system will try to use consecutive device numbers for the devices, starting from zero. However, this is not a requirement, and the user can assign numbers that are not consecutive. However, when running Modsoft or Concept, adapter numbers 0 or 1 must be used.
- **Polling Interval**: This parameter specifies the polling interval in milliseconds that the driver will use when running in the polled mode. The valid range for the polling interval is 20-1000 milliseconds. The default value is 20 ms.
- **Socket Number**: The socket number into which the PC Card is plugged. Socket numbers start with 1.

- **Node Address**: The Modbus Plus node address of the adapter. This address must be unique on the network.
- **Reload Driver**: When this box is checked, the driver will reload using the new configuration parameters after the OK button is clicked.
- **Status/Conflicting Device List**: Contains any error messages from the driver, along with a list of devices that conflict with the current adapter. See *Troubleshooting the MBX Driver* on page 42 for a list of possible error messages.

Finally, the *System Settings Change* dialog may be displayed. Since the MBX Driver is Plug and Play, the system can restart the driver without restarting the system. Select either ***Yes*** or ***No***.



You must restart your computer before the new settings will take effect.

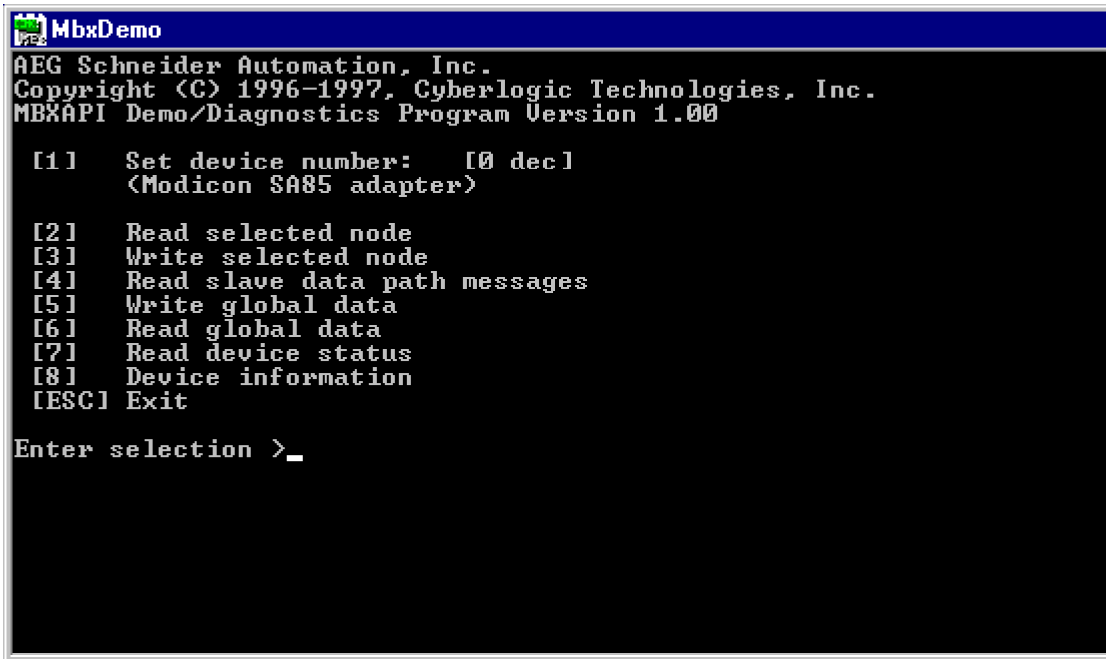
Do you want to restart your computer now?

2.4 Verification of the Driver Operation

After successful installation and device configuration the user can run the 32-bit *MbxDemo* program to verify the operation of the MBX Driver.

Procedure **Running MBX Demo**

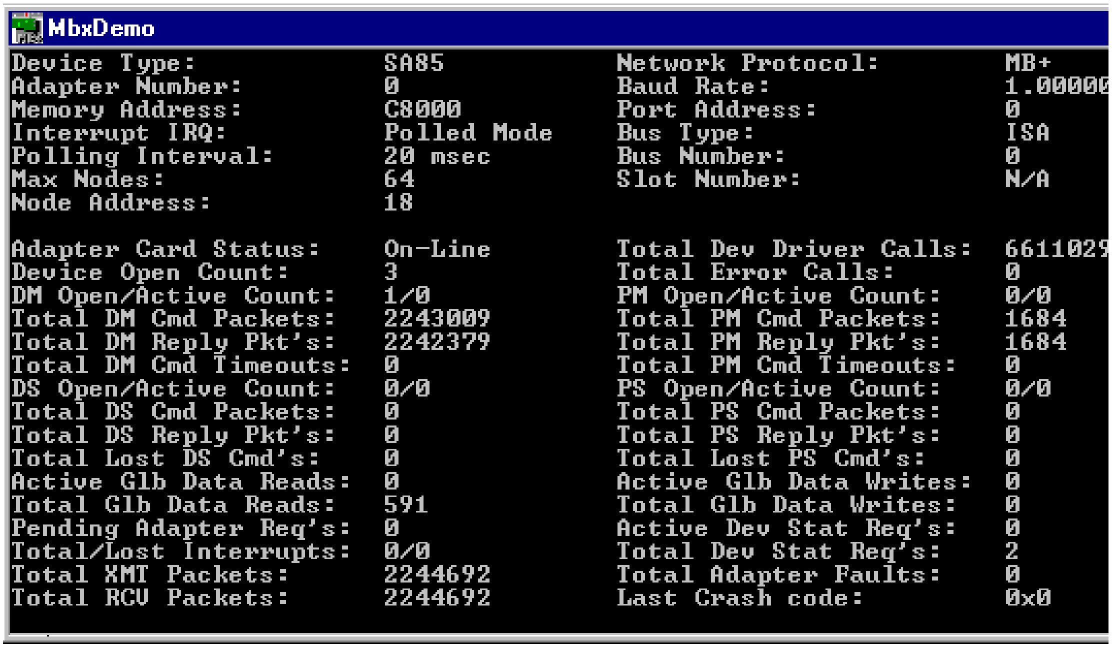
1. The initial screen of the *MbxDemo* program provides a number of selections:



MbxDemo

```
AEG Schneider Automation, Inc.
Copyright (C) 1996-1997, Cyberlogic Technologies, Inc.
MBXAPI Demo/Diagnostics Program Version 1.00

 [1]   Set device number:   [0 dec]
       (Modicon SA85 adapter)

 [2]   Read selected node
 [3]   Write selected node
 [4]   Read slave data path messages
 [5]   Write global data
 [6]   Read global data
 [7]   Read device status
 [8]   Device information
 [ESC] Exit

Enter selection >_
```

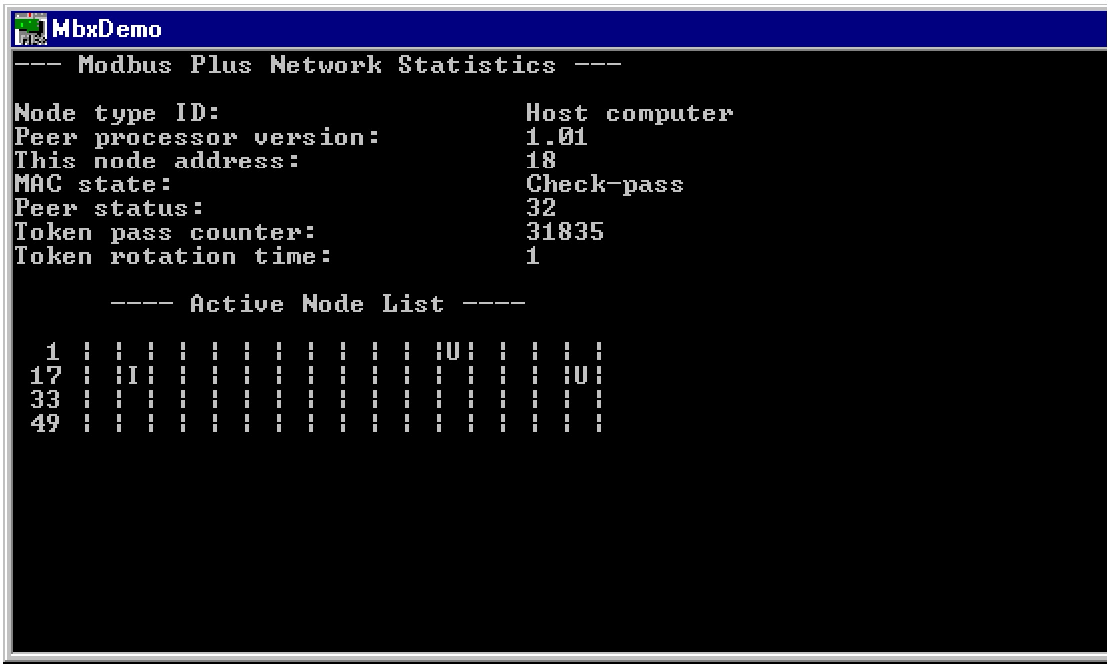
2. Select number ***[8] Device Information***. This screen shows configuration, statistical, and diagnostic information about the driver, the device, and the network.



```
MbxDemo
Device Type:             SA85          Network Protocol:         MB+
Adapter Number:          0             Baud Rate:                1.00000
Memory Address:          C8000         Port Address:             0
Interrupt IRQ:           Polled Mode   Bus Type:                 ISA
Polling Interval:        20 msec       Bus Number:               0
Max Nodes:               64            Slot Number:              N/A
Node Address:            18

Adapter Card Status:     On-Line       Total Dev Driver Calls:   6611029
Device Open Count:       3             Total Error Calls:        0
DM Open/Active Count:    1/0           PM Open/Active Count:     0/0
Total DM Cmd Packets:    2243009       Total PM Cmd Packets:     1684
Total DM Reply Pkt's:    2242379       Total PM Reply Pkt's:     1684
Total DM Cmd Timeouts:   0             Total PM Cmd Timeouts:    0
DS Open/Active Count:    0/0           PS Open/Active Count:     0/0
Total DS Cmd Packets:    0             Total PS Cmd Packets:     0
Total DS Reply Pkt's:    0             Total PS Reply Pkt's:     0
Total Lost DS Cmd's:     0             Total Lost PS Cmd's:      0
Active Glb Data Reads:   0             Active Glb Data Writes:   0
Total Glb Data Reads:    591           Total Glb Data Writes:    0
Pending Adapter Req's:   0             Active Dev Stat Req's:    0
Total/Lost Interrupts:   0/0           Total Dev Stat Req's:     2
Total XMT Packets:       2244692       Total Adapter Faults:     0
Total RCV Packets:       2244692       Last Crash code:          0x0
```

3. Hit ***<Esc>*** to get back to the main menu.

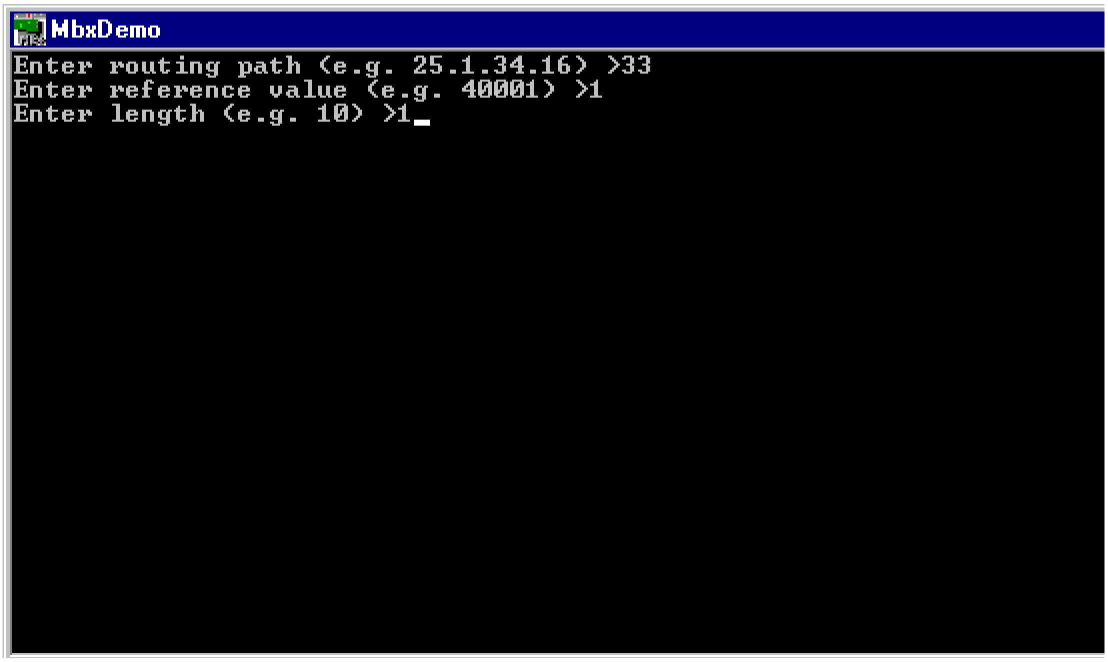
4. Select number ***[7] Read device status***. The following screen will appear. This screen should show all active nodes on the network.



```
--- Modbus Plus Network Statistics ---

Node type ID:                    Host computer
Peer processor version:          1.01
This node address:               18
MAC state:                       Check-pass
Peer status:                     32
Token pass counter:              31835
Token rotation time:             1

      ---- Active Node List ----

  1 ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦U¦ ¦ ¦ ¦ ¦
 17 ¦ ¦I¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦U¦
 33 ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦
 49 ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦
```

5. Hit ***<Esc>*** to get back to the main menu.

6. Select number ***[2] Read selected node***. You should be able to read data from any active node on the network.



MbxDemo

```
Enter routing path (e.g. 25.1.34.16) >33
Enter reference value (e.g. 40001) >1
Enter length (e.g. 10) >1_
```

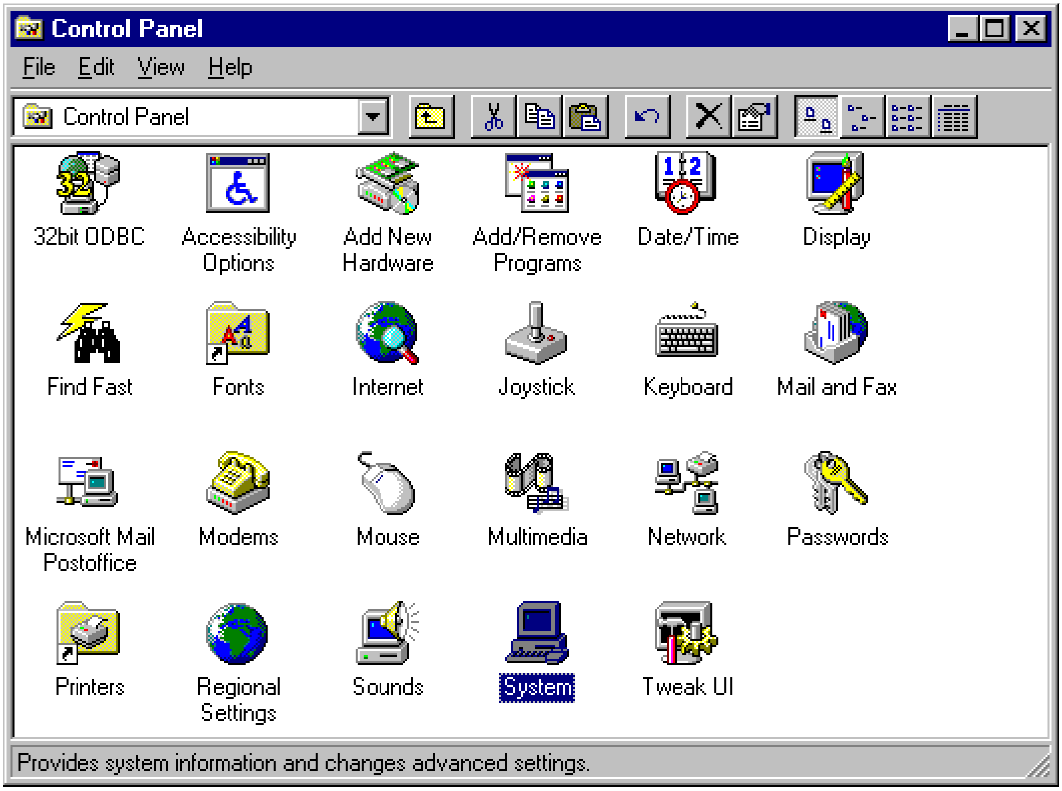
2.5 Troubleshooting the MBX Driver

2.5.1 Viewing Error Messages

During system startup, the MBX Driver may detect a number of configuration problems. When a problem is detected, the driver logs an error message. In order to view the error messages, follow the procedure below.

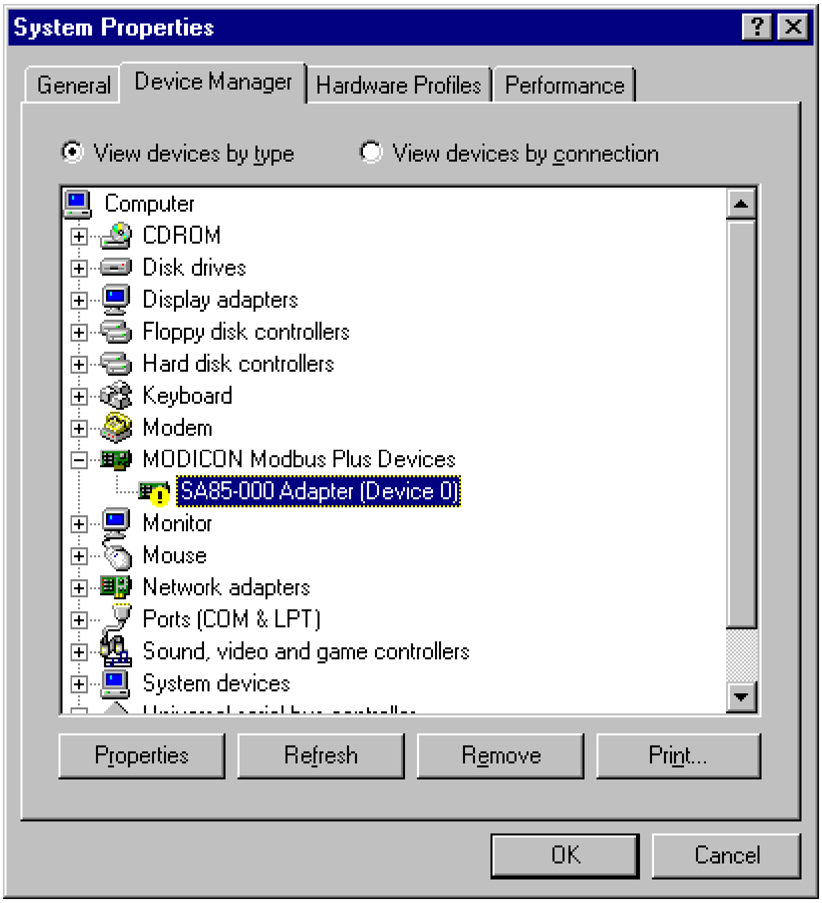
Procedure **Viewing Error Messages**

1. Go to the ***Control Panel*** and double click on the ***System*** icon.

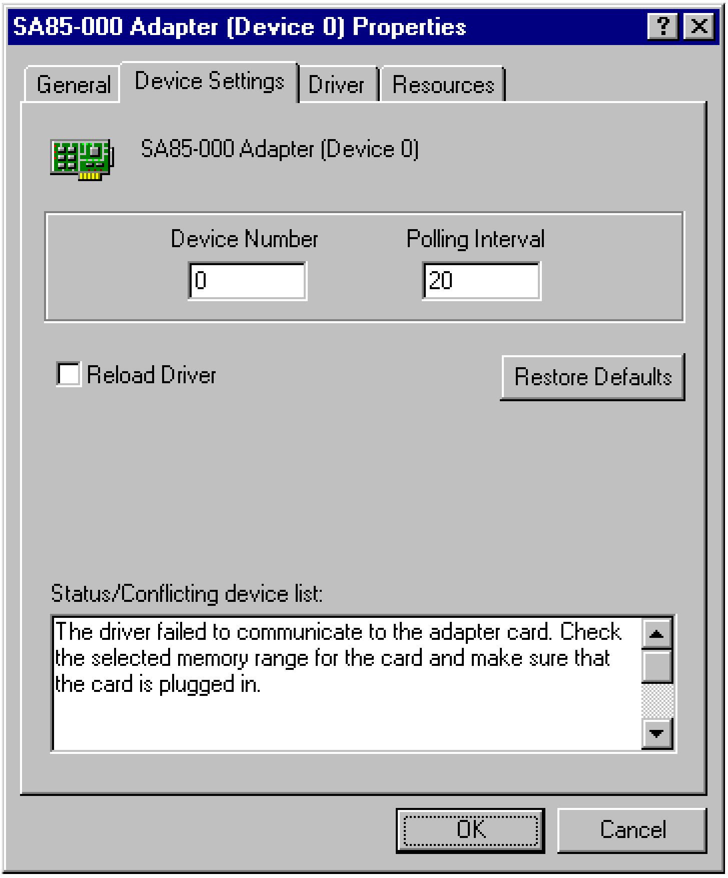


Provides system information and changes advanced settings.

2. Choose the ***Device Manager*** tab. Drill down into the ***MODICON Modbus Plus Devices*** item. Select the device that needs its configuration modified and click the ***Properties*** button.



3. The Status/Conflicting device list box will list the last error message logged by the driver. If there are no error messages, you can run the MbxDemo program provided in the installation directory to verify the driver operation. Refer to *Verification of the Driver Operation* on page 38 for more details.


SA85-000 Adapter (Device 0) Properties

SA85-000 Adapter (Device 0)

Device Number: 0

Polling Interval: 20

Reload Driver: [ ]

Status/Conflicting device list:

The driver failed to communicate to the adapter card. Check the selected memory range for the card and make sure that the card is plugged in.

2.5.2 MBX Driver Messages

Errors

- Adapter card failed hardware reset.

  Check conflicting device list for possible conflicts with other devices in the system. Make sure that the adapter card is plugged in. Make sure that the adapter card is properly configured. Try a different adapter card.

- Adapter card failed to execute <command name> command.

  Check conflicting device list for possible conflicts with other devices in the system. Try a different adapter card.

- Adapter card memory window allocation error.

  The system ran out of the resources to allocate memory window for the PCMCIA card. Remove other adapter cards in the system that may free up the required memory.

- Adapter card's dual-port memory diagnostics at selected memory address failed.

  Check conflicting device list for possible conflicts with other devices in the system. For ISA bus adapters, make sure that the bus speed is set to 8 MHz. Some systems with ISA bus do not allow mixing 8-bit and 16-bit cards in the same 128K memory page (like C000-D000). Remove 16-bit cards that may interfere with your adapter card. Try a different adapter card.

- Adapter card's interface diagnostics failed.

  Check conflicting device list for possible conflicts with other devices in the system. For ISA bus adapters, make sure that the bus speed is set to 8 MHz. Some systems with ISA bus do not allow mixing 8-bit and 16-bit cards in the same 128K memory page (like C000-D000). Remove 16-bit cards that may interfere with your adapter card. Try a different adapter card.

- Duplicate adapter number detected.

  Two devices have the same device number. The Status/Conflicting device list should name the conflicting devices. Change the device number for the conflicting devices, making sure all devices have unique device numbers. *Editing Current Configuration* on page 19 has information on changing a device's configuration.

- Error reading parameter <parameter name>.

  First check the device configuration and reload the driver. If the error still occurs, go into the ***Control Panel/System/Device Manager*** and remove the device. Then create the device again. See *MBX Driver Configuration* on page 11 for information on both adding a device and changing a device's configuration.

- Interrupt initialization error. Check conflicting device list for possible conflicts with other devices in the system.

  Check the Status/Conflicting device list found on the device's ***Resources*** page. The conflicting device should be listed. Either select a new interrupt or put the device into polled mode. Refer to *Editing Current Configuration* on page 19 for information about changing the configuration.

- Invalid value for <parameter name> (<parameter value>).

  An invalid configuration parameter was specified. Refer to *Editing Current Configuration* on page 19 for information about changing the configuration.

- MapPhysToLinear failed.

  Your system may be unstable. Reinstall Windows 95.

- Out of memory error in <function name>.

  Free up some memory by closing any unnecessary applications or add more memory to the system.

- PCMCIA Card Services map memory page failed.

  Make sure that the adapter card is plugged into selected PCMCIA socket. (First socket is Socket Number 1). Try different adapter card.

- PCMCIA Card Services not present error. Go to Control Panel and select PC Card (PCMCIA) to active the PCMCIA Card Services.

  Go to ***Control Panel*** and select ***PC Card (PCMCIA)*** to activate the PCMCIA Card Services.

- PCMCIA Card Services registration failed.

  Go to ***Control Panel*** and select ***PC Card (PCMCIA)*** to activate the PCMCIA Card Services.

- PCMCIA Card Services window request failed. Make sure that the adapter card is plugged into selected PCMCIA socket. (First socket is Socket Number 1).

  Make sure that the adapter card is plugged into selected PCMCIA socket. (First socket is Socket Number 1). Try different adapter card.

- The driver failed to communicate to the adapter card. Check the selected memory range for the card and make sure that the card is plugged in.

  Check conflicting device list for possible conflicts with other devices in the system. Make sure that the adapter card is plugged in. Make sure that the adapter card is properly configured. Try different adapter card.

- Unexpected error in <function name>. Please contact technical support of manufacturer.

  Indicates a programming bug in the device driver.

Informational

- This is a promotional copy of the device driver. It will operate for 4 hours.

Crash Codes

Occasionally, due to adapter card malfunctions, the MBX Driver may detect an adapter card fault. In most cases this would be due to electrical interference (both internal and external to computer system) but it can also be an indication of genuine card failure. The MBX Driver tries to automatically recover from these failures. Every time a fault condition is detected an internal adapter fault counter is incremented as well as the last *crash code* is internally recorded. Both of these numbers can be viewed through the ***Device Information*** screen of the *MbxDemo* program. The following is a complete list of all crash codes that can aid in diagnosing these types of problems.

Crash	Symbolic Name	Error Type	Description
0x00	IFCINTACT	None	Interface operational
0x01	IFCTIMOUT	Interface	2.0-sec interface timeout
0x02	BADIFCOPC	Interface	Bad interface op-code
0x03	IFCDATERR	Interface	Interface data error
0x04	IFCTSTERR	Interface	Interface test error
0x05	IFCDONERR	Interface	Interface x-fer done error
0x06	BADIFCPTH	Interface	Bad interface path
0x07	BADXFRSVR	Interface	Bad transfer state
0x08	BADXFRLEN	Interface	Bad transfer length
0x09	GLBDATLEN	Interface	Global-data length error
0x0A	GLBDATADR	Interface	Global-data address error
0x0B	GLBDATPRS	Interface	Global-data not present
0x81	CKSUMERR	Fatal	PROM check-sum error
0x82	RAMDATERR	Fatal	Internal RAM data test error
0x83	EXTDATERR	Fatal	External RAM data test error
0x84	EXTADRERR	Fatal	External RAM address test error
0x85	BADCTINDX	Fatal	Bad confidence test index
0x86	EXT0EVENT	Fatal	External *int0* event error
0x87	EXT1EVENT	Fatal	External *int1* event error
0x88	DMA0EVENT	Fatal	DMA *int0* event error
0x89	COMMEVENT	Fatal	Comm-int event error
0x8A	XMTNGEVNT	Fatal	Xmit-no-good event error
0x8B	RSPTOSVAR	Fatal	No-response timeout MAC-state
0x8C	RSPTOIDLE	Fatal	No- response timeout MAC-idle
0x8D	RCVOKSVAR	Fatal	Receive-ok MAC-state
0x8E	XMTOKSVAR	Fatal	Transmit-ok MAC-state

Crash	Symbolic Name	Error Type	Description
0x8F	NORCVBUF	Fatal	No receive buffer free
0x90	BADINXLEN	Fatal	Bad input-transfer length
0x91	RESBUFERR	Fatal	Reserved rcv-buf error
0x92	BADTCSVAR	Fatal	Bad trans-control state
0x93	BADWRKREQ	Fatal	Bad work request bit
0x94	OVFDATQUE	Fatal	Node-queue overflow
0x95	BADDATQUE	Fatal	Bad data-queue overflow
0x96	NOPATHERR	Fatal	Empty data-path error
0x97	BADPTHINX	Fatal	Bad path search index
0x98	BADDSPATH	Fatal	Bad data-slave path
0x100			Uncontrolled adapter crash
0x101			Adapter card initialization fault
0x102			Adapter card software reset fault
0x103			Adapter I/O timeout fault

2.6 Frequently Asked Questions

Helpful Hints

- We suggest that after you install the MBX Driver software, you run the *MbxDemo* program to ensure that the driver is configured and running properly. See *Verification of the Driver Operation* on page 38 for more details.
- If you are experiencing problems with performance, make sure both the adapter card and the driver are set up for *polled mode* or both are set up for *interrupt mode*.
- In polled mode the recommended rate to use for optimum performance is 20 millisecond polling rate. The interrupt mode of operation will provide higher message rate at the expense of higher CPU load. Low-end systems (such as 486 based systems) may provide a better overall performance with adapter cards configured to run in polled mode.
- Make sure there are no address conflicts with the board address. The *Windows 95 Diagnostics* application that comes with Windows 95 (found in the *Administrative Tools* group) may aid in detecting hardware conflicts.
- Make sure that you are communicating through the right device.
- Make sure that you selected a unique node address for your device.

Frequently Asked Questions

- I've installed the software. What's next?

  The next step is to configure a device. You'll need to know the memory address and interrupt, if any, the card will use. See *MBX Driver Configuration* on page 11 for more details.

- I've configured my device, but when the system boots up and I look in the Device Manager, the device has an exclamation point on it. What does that mean?

  An error has occurred during the loading of the driver. Check the Status/Conflicting device list to see what error occurred. See *Troubleshooting the MBX Driver* on page 42 for help in correcting the problem.

- The Status/Conflicting device list has some error messages. How do I fix that?

  The two most common errors are the results of either a conflict with another device or the driver configuration not matching the card configuration. Verify that the card's memory address matches the address the driver is using. Also compare the state of the interrupt jumper on the card with the polled mode/interrupt mode setting of the driver.

If the configurations match, there may be a conflict in the system or the card may be faulty. If possible, try a card that is known to be good in the system with the same settings. If errors still occur, try setting the card to polled mode and moving to a new memory address (C8000, D0000, D4000, and D8000 are usually good addresses to try). Make sure you change both the driver and card settings.

- There might be a conflict with my device. What do I do?

  Try setting the card to polled mode and moving to a new memory address (C8000, D0000, D4000, and D8000 are usually good addresses to try). Make sure you change both the driver and card settings.

- When I configure a device, should I use polled mode or interrupt mode?

  We recommend polled mode. Interrupt mode gives you slightly higher performance, but it puts a greater load on the CPU. Also, you then have to find a free interrupt and worry about interrupt conflicts. For the majority of applications, running in polled mode with a 20 millisecond polling interval will provide sufficient throughput. Whichever mode you choose, make sure the jumper setting on the card matches the driver setting.

- The card seems to be working, but I can’t see one of the nodes on the network. What’s wrong?

  There are two things to check. First, make sure that the card is plugged into the network. Second, it’s likely that both nodes have the same network node address. Shutdown the system, change the card’s network address by changing the DIP switch settings (refer to the *"Modicon IBM Host Based Devices User’s Guide"* from Modicon), and then restart the system. You should now be able to see all the nodes.

- I have two devices in the system. How do I communicate through the second one?

  MbxDemo uses the device number to determine which card to use. Option 1, *Set Device Number,* lets you choose which device the demo will use. If you are using some other software, contact the manufacturer for more information on using multiple cards.

Virtual MBX Driver

3

3.1 Introduction

The Virtual MBX Driver for the Microsoft Windows 95 operating system allows all existing 16-bit DOS or Windows 3.x NETLIB/NetBIOS-compatible applications to run under Windows 95 in their original binary form. This includes programs such as ModSoft, ModLink, MBPSTAT, and hundreds of other custom applications written by software developers and system integrators. All of these applications can now run concurrently with other 32-bit applications, sharing the same 32-bit device driver.

For DOS applications and Win16 applications that do not use the MBPLUS.DLL library, the Virtual MBX Driver fully emulates the operation of the old SA85.SYS and MBPHOST.SYS drivers from Schneider Automation. Therefore, the SA85.SYS and MBPHOST.SYS drivers are no longer needed and should not be used. For Winb16 applications that use the MBPLUS.DLL, a new version of that library is provided. It is strongly recommended that this new version of the library is used in order to achieve the best performance.

The Virtual MBX Driver fully supports all NETLIB/NetBIOS features. This includes support for *Data Master/Slave*, *Program Master/Slave* and *Global Data*. The Virtual MBX Driver uses the MBX Driver, and its high performance 32-bit kernel mode device driver, for all of its local I/O functions. It can also use the Remote MBX Driver for network based installations. (The MBX Driver and Remote MBX Driver are purchased separately and either the MBX Driver or the Remote MBX Driver is required for the operation of the Virtual MBX Driver.)

The Virtual MBX Driver provides an excellent bridge for users who are porting their applications to Windows 95 but are not ready to abandon their favorite 16-bit applications. Once the driver is installed, you can execute 16-bit programs exactly the same way as in their original environment. In addition, you can concurrently execute multiple instances of the same applications. For example, you can run multiple instances of MBPSTAT.EXE and monitor the operation of multiple Modbus Plus networks.

3.2 Installation

Procedure

Windows 95 Installation Procedure

1. Exit all applications that are using MBX products like the *MBX Driver*, the *Remote MBX Driver*, or the *Virtual MBX Driver* before installing the new version. If the Virtual MBX Driver is running, stop it before beginning the installation.

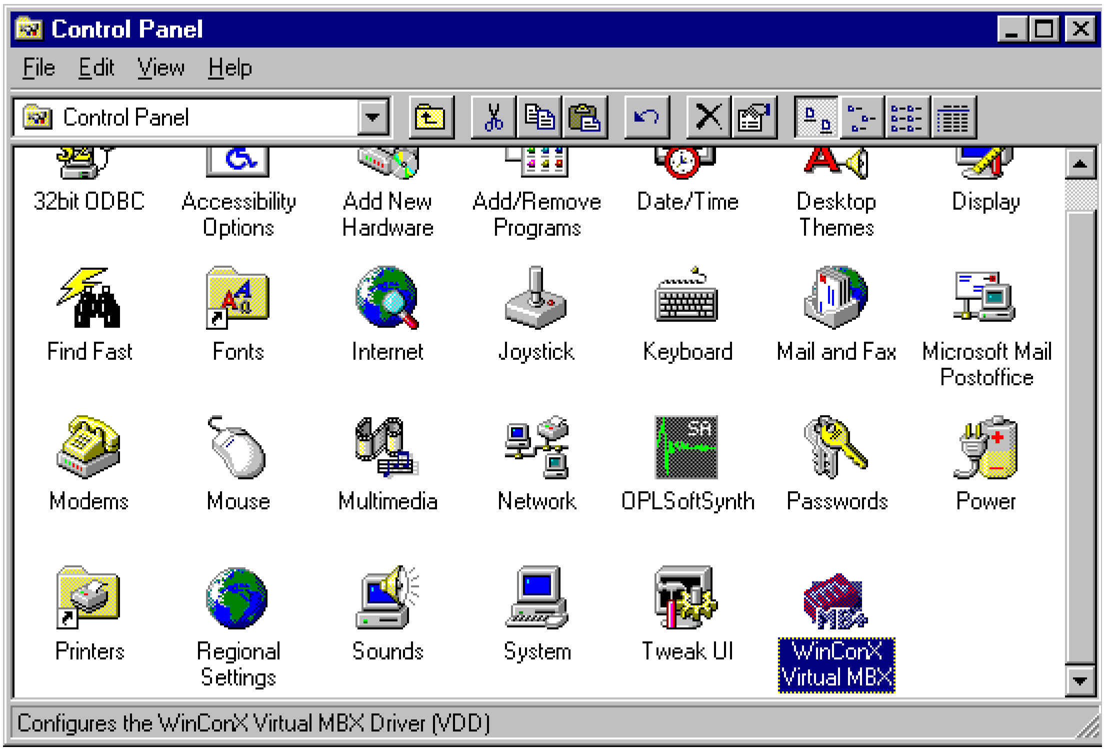
2. Place the installation disk in the floppy drive (either Drive A: or B:).

3. Go to the *Control Panel*, select the ***Add/Remove Programs*** icon, and click the ***Install*** button. click ***Next>*** and then ***Finish*** to install the software onto the hard drive.

4. Configure the Virtual MBX Driver. See *Virtual MBX Driver Configuration* on page 53 for more information.

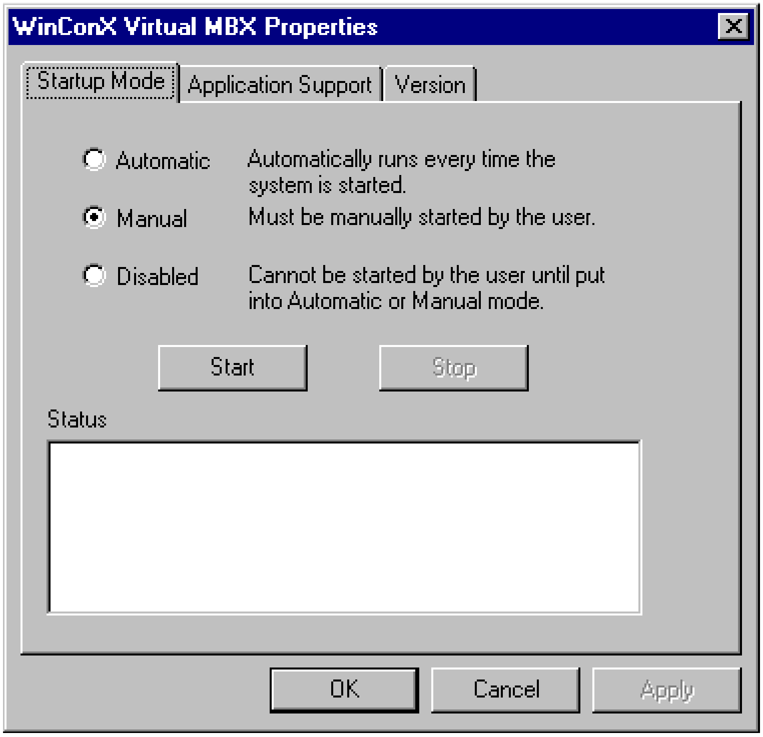
3.3 Virtual MBX Driver Configuration

Before any 16-bit application can run, the Virtual MBX Driver must be configured and started. During the driver installation, a new applet was added to the *Control Panel*. The *WinConX Virtual MBX* applet is used to configure and manually start and stop the Virtual MBX Driver.



Configures the WinConX Virtual MBX Driver (VDD)

3.3.1 Startup Mode



WinConX Virtual MBX Properties

Startup Mode | Application Support | Version

Automatic — Automatically runs every time the system is started.

[✓ Manual] — Must be manually started by the user.

Disabled — Cannot be started by the user until put into Automatic or Manual mode.

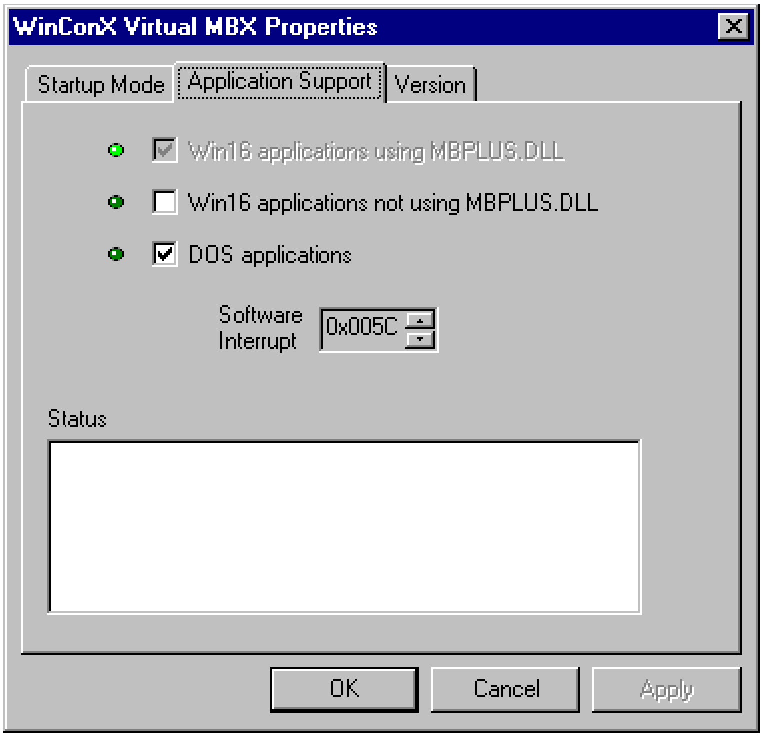
Start | Stop

Status

OK | Cancel | Apply

- **Automatic**: The driver automatically runs whenever the system starts up.
- **Manual**: The driver must be manually started by the user before any 16-bit applications are run.
- **Disabled**: The driver cannot be started (either manually or automatically) until it is put into either Manual or Automatic mode.
- **Start**: Manually starts the driver.
- **Stop**: Manually stops the driver.
- **Status**: Displays status and error messages from the driver.

3.3.2 Application Support



[✓] Win16 applications using MBPLUS.DLL

[ ] Win16 applications not using MBPLUS.DLL

[✓] DOS applications

Software Interrupt: 0x005C

Status

- Win16 applications using MBPLUS.DLL: Many 16-bit Windows application, like Modicon's ModLink, use a DLL named MBPLUS.DLL to communicate with the Modbus Plus device. During the Virtual MBX Driver installation, a new version of this DLL is copied into the Windows' SYSTEM directory. If this box is checked (and shown grayed), a copy of the new MBPLUS.DLL was found in the Windows' SYSTEM directory.

CAUTION

DLL MAY BE OVERWRITTEN

Some applications may overwrite this DLL with an older, Windows 3.x version. If this happens, check this box. The next time the Virtual MBX Driver is started, the new MBPLUS.DLL will be copied into the Windows' SYSTEM directory.

Failure to observe this precaution can result in injury or equipment damage.

- **Win16 applications not using MBPLUS.DLL**: Some 16-bit Windows applications, like Modicon’s MSL, do not communicate using MBPLUS.DLL. Checking this box provides support for these types of applications.

+ **Note:** Supporting these applications incurs some system overhead. You will get better system performance if the applications you are using work properly without this support.

- **DOS applications**: Provides support for DOS applications like ModSoft and MBPSTAT.
- **Software Interrupt**: The interrupt through which the application will communicate with the device. (The new version of the MBPLUS.DLL does not require this interrupt.)
- **Status Lights**: Next to each selection is a status light. These status lights indicate the following conditions.
  - **Bright Green Light:** The selection is active and working correctly.
  - **Dark Green Light:** The selection is currently not active.
  - **Exclamation Icon:** The selection is not working correctly and, as a result, is not active. Refer to the status box for possible error messages.
- **Status**: Displays status and error messages from the driver.

3.4 Frequently Asked Questions

- Can I run multiple 16-bit DOS/Windows programs at the same time?

  Yes, you can.
- I stopped the Virtual MBX Driver from the Control Panel applet but I see that I can still run ModLink. Why is that?

  Most 16-bit Windows applications, like Modicon’s ModLink, use a DLL named MBPLUS.DLL to communicate with the MB+. During the Virtual MBX Driver installation, a new version of this DLL is copied into the Windows’ SYSTEM directory. This DLL is always active and does not require the rest of the Virtual MBX to be started.

Remote MBX Driver

4

4.1 Introduction

The Remote MBX Driver for Microsoft Windows 95 operating system provides remote connectivity for applications running on client nodes to access Modbus Plus networks from remote locations via standard LANs.

The Remote MBX Driver provides access to remote client nodes over any Windows 95-compatible computer network, enabling the remote client nodes to access the (configured) MBX Driver devices residing on server nodes running the MBX Remote Server (part of the MBX Driver for Windows NT, purchased separately). However, a host interface adapter, such as Schneider Automation’s SA85 card, is not required on the client node. The Remote MBX Driver provides complete MBX functionality to the client node, including support for *Data Master/Slave*, *Program Master/Slave* and *Global Data*. Any node on the network can be configured as a client to a number of remote servers and at the same time communicate to its local Modbus Plus networks.

The Remote MBX Driver provides compatibility with applications written to utilize its high performance application programming interface (API) as well as the industry standard NETLIB interface specification from Schneider Automation. The 32-bit NETLIB compatibility provides an excellent bridge for developers who would like to port their 16-bit, NETLIB-compatible applications to Windows 95. Developers of new applications can use either the NETLIB or the high performance MBXAPI programming interface. To obtain the MBXAPI software development kit, including the MBXAPI specification, MBXAPI sample source code, and NETLIB sample source code, contact Schneider Automation or Cyberlogic. For complete reference of all NETLIB library functions refer to "*Modicon IBM Host Based Devices User’s Guide*" from Schneider Automation (Order# 890 USE 102 00).

Running 16-Bit Software

A companion product, the Virtual MBX Driver, will allow all 16-bit NETLIB/NetBIOS-compatible applications (e.g., ModSoft) to run concurrently with all 32-bit applications in the same computer. The Virtual MBX Driver will allow multiple 16-bit applications as well as multiple instances of a single 16-bit application to run under the 32-bit Windows 95 operating system. (The Virtual MBX Driver is purchased separately and it requires either the MBX Driver or the Remote MBX Driver to operate).

4.2 Installation

Procedure

Windows 95 Installation Procedure

1. Exit all applications that are using MBX products like the *MBX Driver*, the *Remote MBX Driver*, or the *Virtual MBX Driver* before installing the new version. If the Virtual MBX Driver is running, stop it before beginning the installation.

2. Place the installation disk in the floppy drive (either Drive A: or B:).

3. Go to the *Control Panel*, select the ***Add/Remove Programs*** icon, and click the ***Install*** button. Click ***Next>*** and then ***Finish*** to install the software onto the hard drive.

4. Configure the Remote MBX Driver. See the *Remote MBX Driver Configuration* on page 59 for more information on configuring devices.

4.3 Remote MBX Driver Configuration

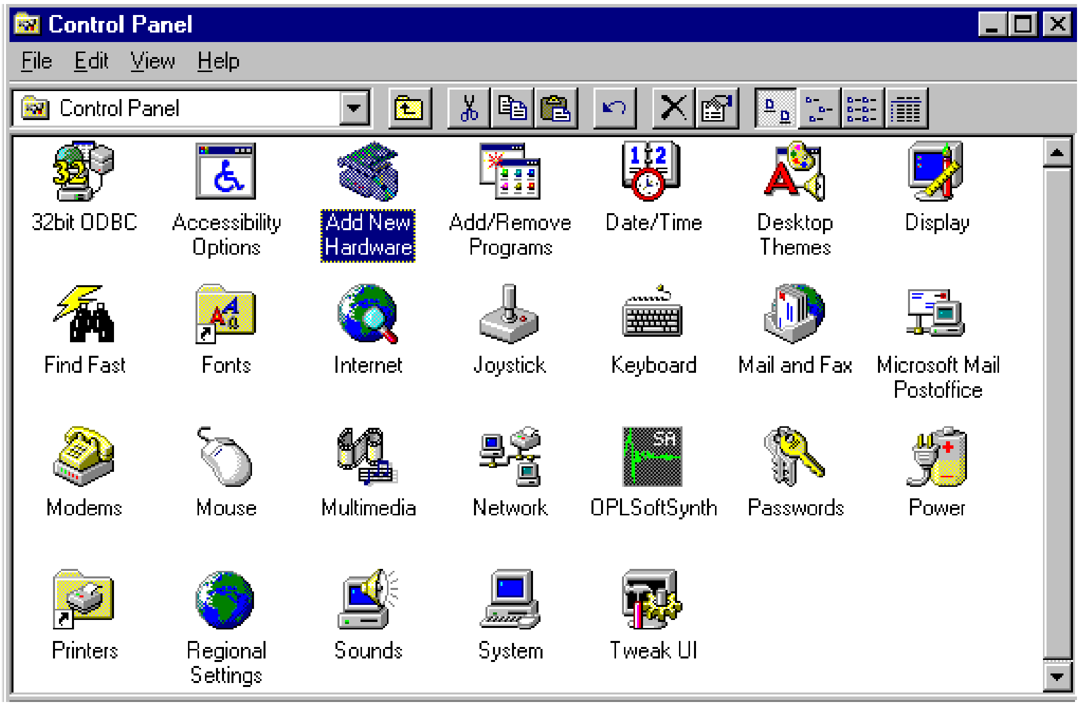
The Remote MBX Driver is designed to operate in remote client nodes without a need for physical host interface adapters. Therefore, the Remote MBX Driver configuration involves creation and configuration of logical devices (as opposed to physical devices, such as an SA85 card). The configuration of the Remote MBX Driver is similar to the configuration of the MBX Driver, except where the MBX Driver deals with physical host interface adapters, the Remote MBX Driver will refer to logical devices.

The Remote MBX Driver requires a number of parameters to be set for each device. The following sections describe the process for adding new adapters and modifying the configurations of existing adapters.

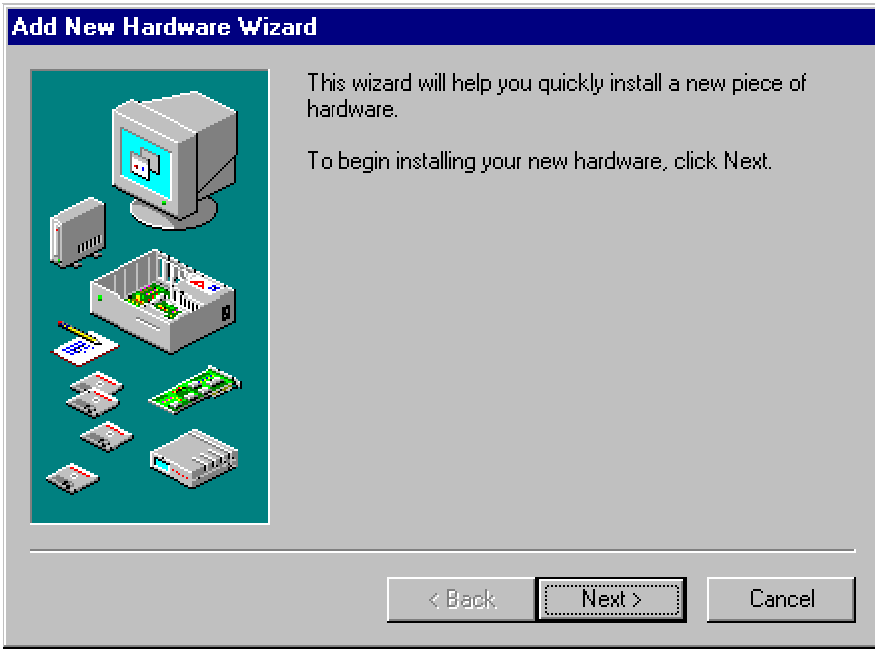
4.3.1 First Time Configuration

Procedure **First Time Configuration**

1. Go to the ***Control Panel*** and double click on the ***Add New Hardware*** icon.



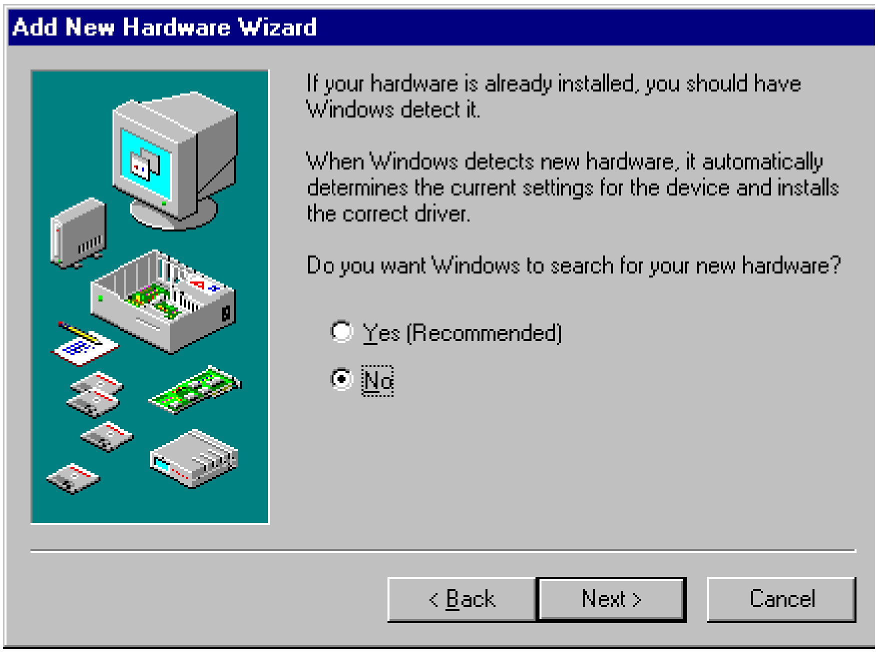
2. Click *Next >* when the following dialog appears.



This wizard will help you quickly install a new piece of hardware.

To begin installing your new hardware, click Next.

3. When asked if Windows should search for the new hardware, select ***No*** and click ***Next >***.



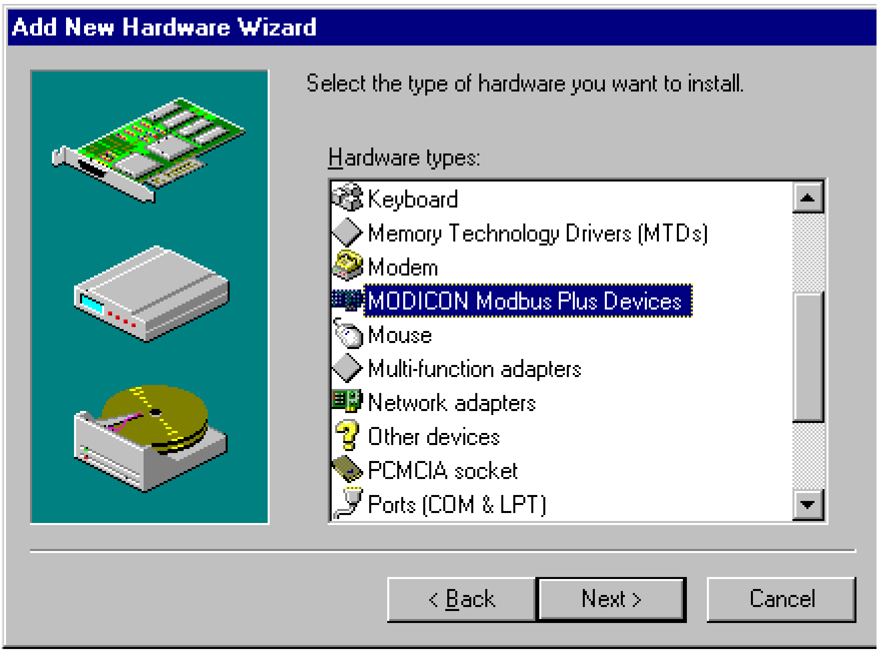
If your hardware is already installed, you should have Windows detect it.

When Windows detects new hardware, it automatically determines the current settings for the device and installs the correct driver.

Do you want Windows to search for your new hardware?

Yes (Recommended) [✓ No]

4. For the hardware type, select ***MODICON Modbus Plus Devices***.



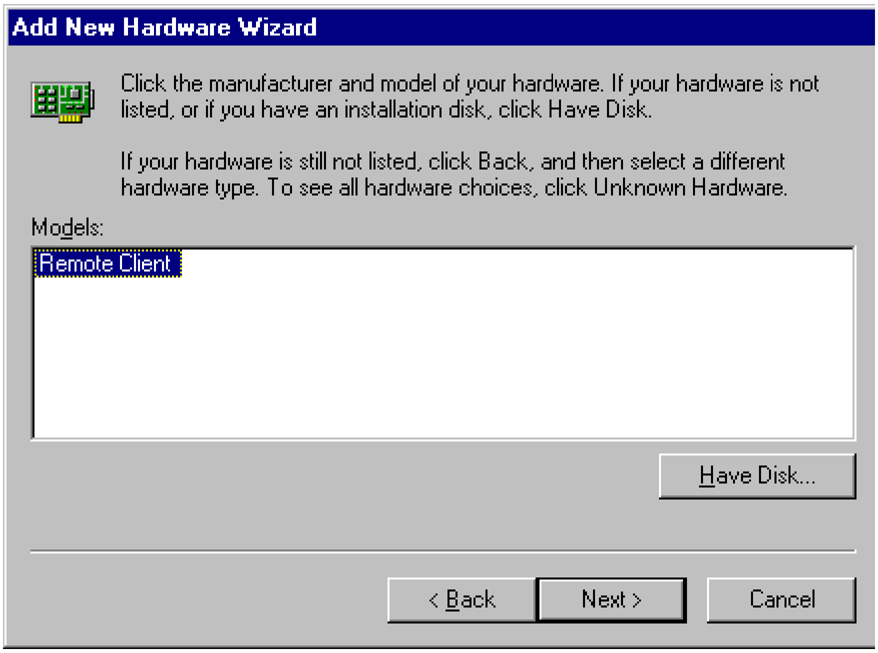
Add New Hardware Wizard

Select the type of hardware you want to install.

Hardware types:

- Keyboard
- Memory Technology Drivers (MTDs)
- Modem
- MODICON Modbus Plus Devices
- Mouse
- Multi-function adapters
- Network adapters
- Other devices
- PCMCIA socket
- Ports (COM & LPT)

5. Select the device type from the list that matches the hardware that is installed in the system. Click ***Next >***.


Add New Hardware Wizard

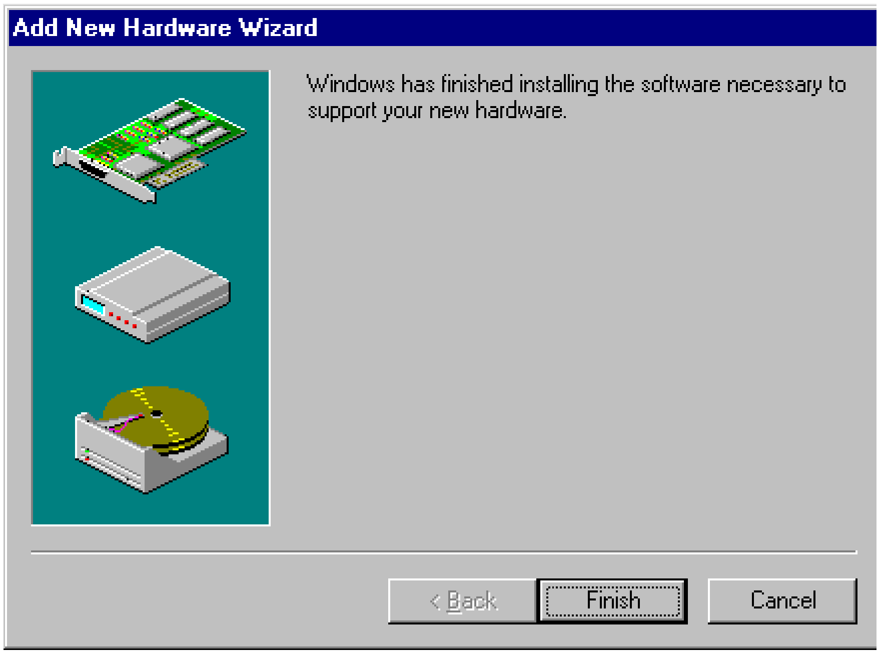
Click the manufacturer and model of your hardware. If your hardware is not listed, or if you have an installation disk, click Have Disk.

If your hardware is still not listed, click Back, and then select a different hardware type. To see all hardware choices, click Unknown Hardware.

Models:

Remote Client

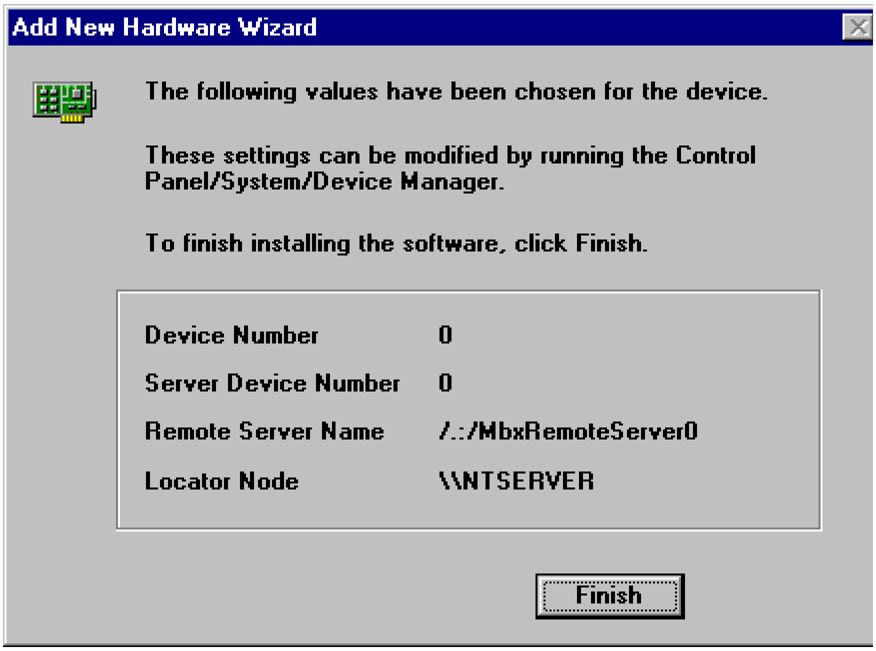
6. Select ***Finish***.



Add New Hardware Wizard

Windows has finished installing the software necessary to support your new hardware.

7. Note the device number selected by the system may not match the value shown in the following figure. See *Editing Current Configuration* on page 67 for information on changing the device settings. Click ***Finish***.



Add New Hardware Wizard

The following values have been chosen for the device.

These settings can be modified by running the Control Panel/System/Device Manager.

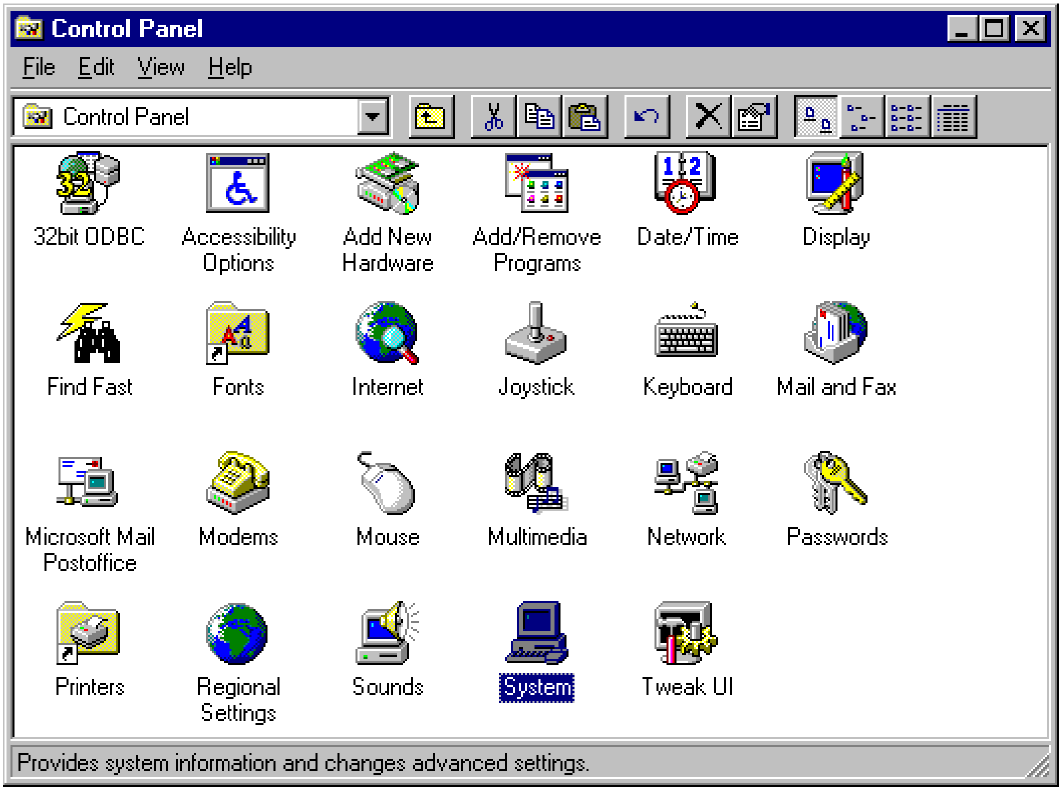
To finish installing the software, click Finish.

Device Number	0
Server Device Number	0
Remote Server Name	/.:/MbxRemoteServer0
Locator Node	\\NTSERVER

4.3.2 Editing Current Configuration

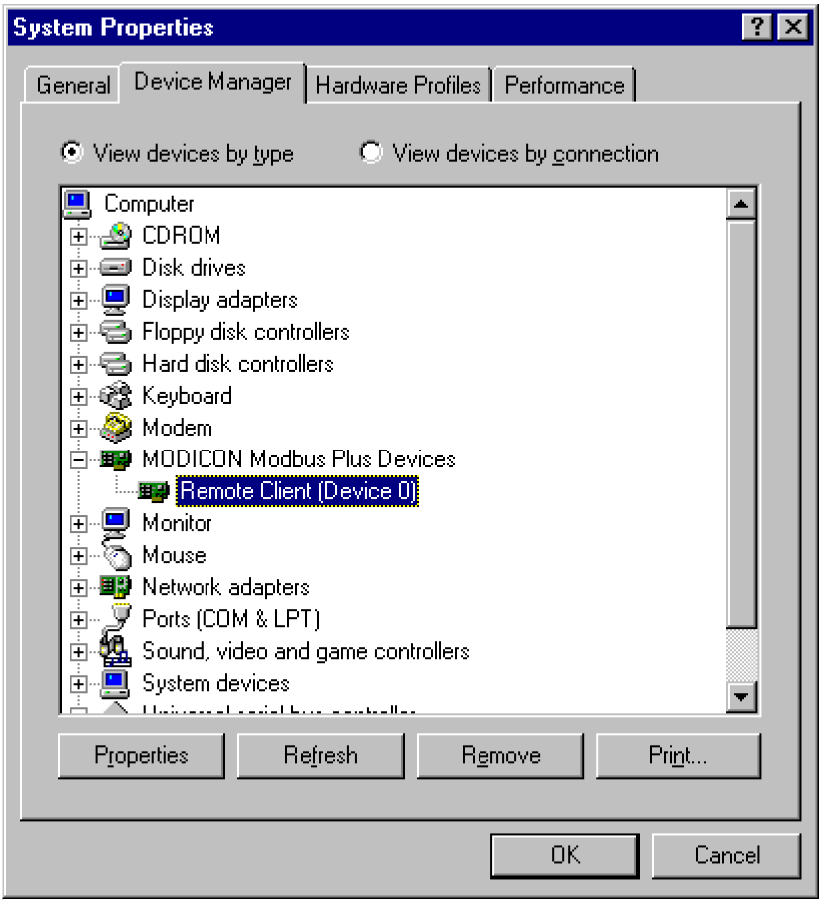
Procedure **Editing Current Configuration**

1. Go to the ***Control Panel*** and double click on the ***System*** icon.



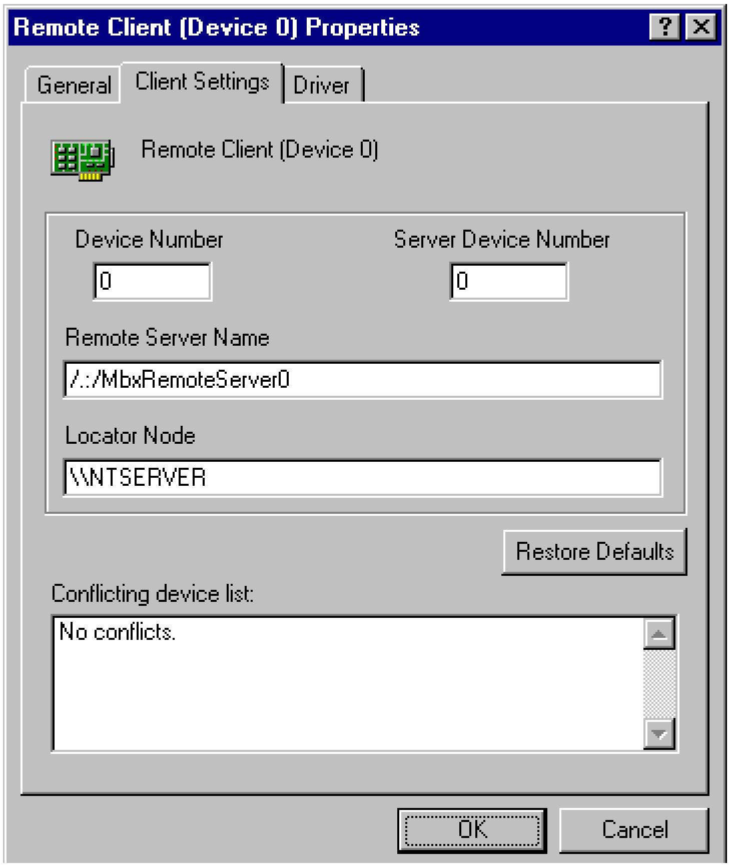
Provides system information and changes advanced settings.

2. Choose the ***Device Manager*** tab. Drill down into the ***MODICON Modbus Plus Devices*** item. Select the device that needs its configuration modified and click the ***Properties*** button.



3. At this point, a property sheet is displayed. See the following sections for specific configuration information about each device type.

4.3.3 MBX Remote Client


Remote Client (Device 0) Properties

Remote Client (Device 0)

Device Number: 0

Server Device Number: 0

Remote Server Name: /.:/MbxRemoteServer0

Locator Node: \\NTSERVER

Conflicting device list:

No conflicts.

- **Device Number**: A number that you assign to every device installed in your system. **This is not the Modbus Plus node address!** By default, the system will try to use consecutive device numbers for the devices, starting from zero. This is not a requirement, and the user can assign numbers that are not consecutive. However, when running Modsoft or Concept, adapter numbers 0 or 1 must be used.
- **Server Device Number**: Designates which device on the server the client will use. As an example, assume a server has two host adapters: Device 0 – SA85, device 1 – AT984. If the client is configured to use Server Device Number 0, the server's SA85 card will be accessed. Using Server Device Number 1 accesses the server's AT984.

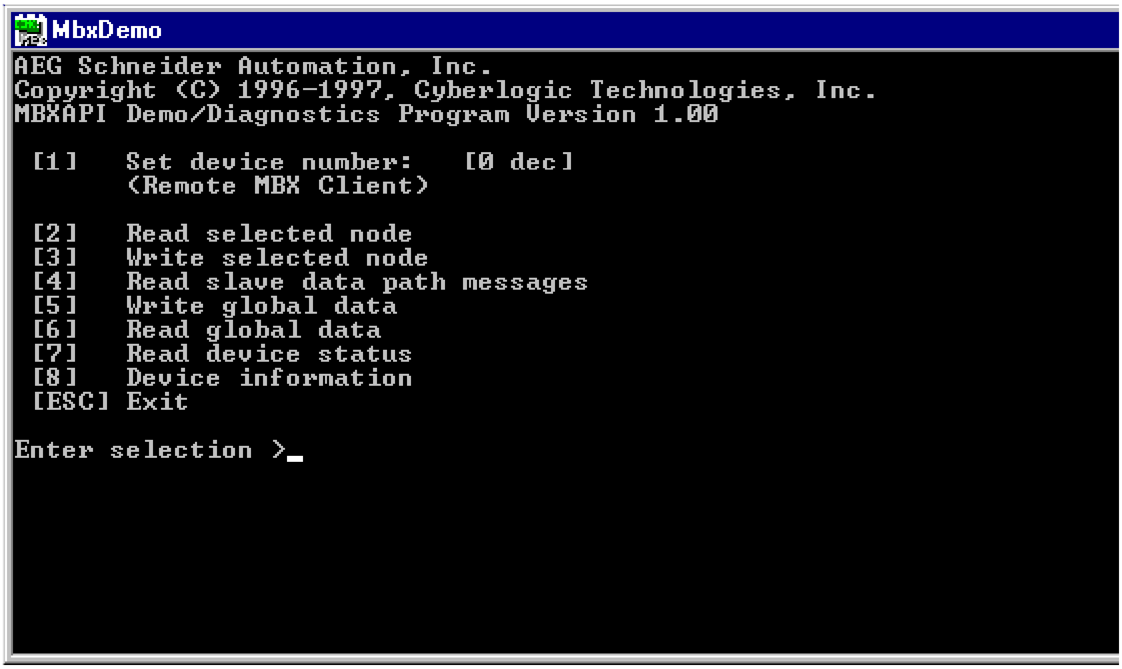
- **Remote Server Name**: The name of the MBX Remote Server with which this client should connect. The name can take one of two forms. The first form (/.:/ServerName) is used when the server and client are in the same workgroup or domain. The second form (/…/ServerDomainOrServerWorkgroupName/ServerName) is used when the server and client are in different workgroups or domains.
- **Locator Node**: Due to a limitation of Windows 95, a remote client requires at least one network node to be running Windows NT. That Windows NT node will find the location of Remote MBX server for the client when a remote connection is made. The name takes the form **ll*NTNodeName***, and will be used by all the remote clients running from a computer. The same locator node can be used by remote clients running on other computers.

4.4 Verification of the Driver Operation

After successful installation and device configuration, the user can run the 32-bit *MbxDemo* program to verify the operation of the Remote MBX Driver.

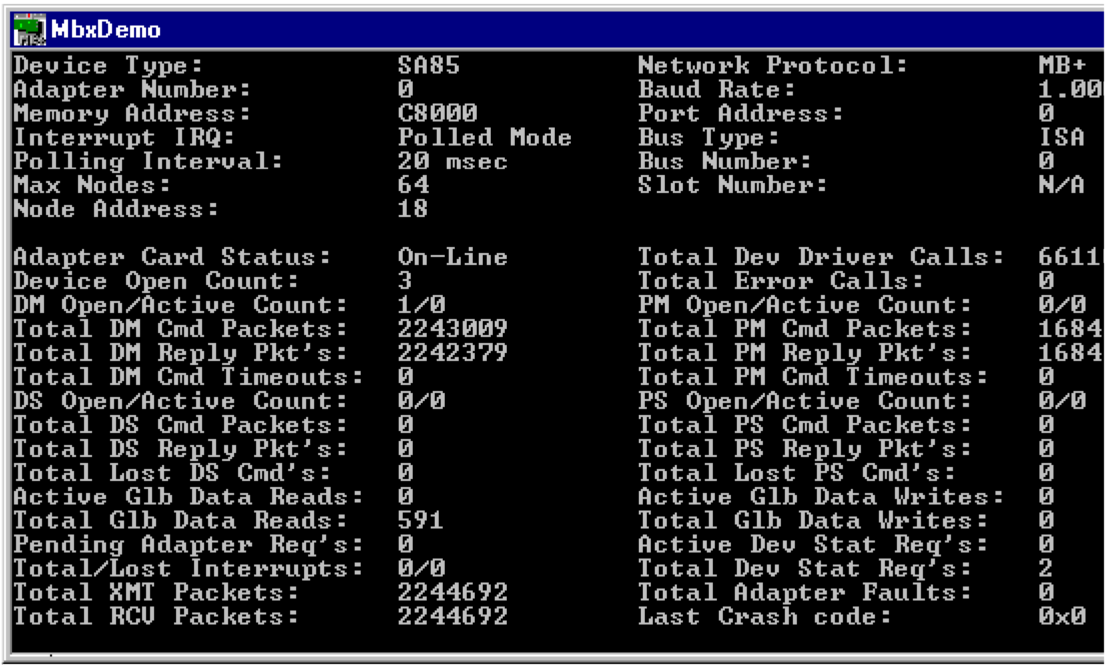
Procedure **Verifying the Operation of the Remote MBX Driver**

1. The initial screen of the *MbxDemo* program provides a number of selections. Select number ***[8] Device Information***.



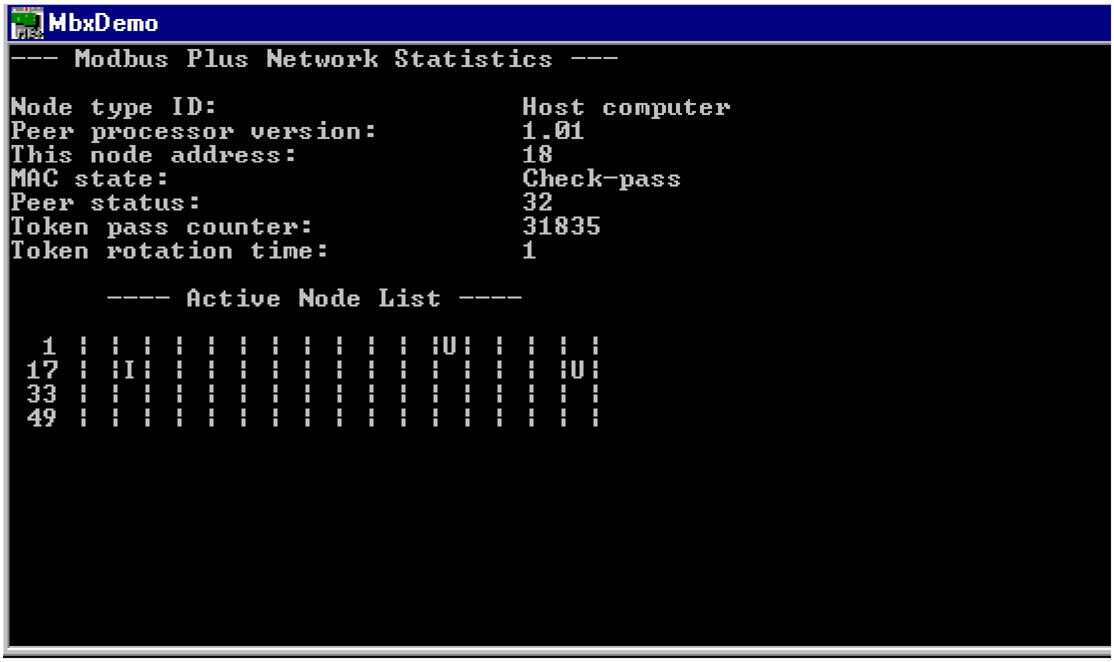
```
AEG Schneider Automation, Inc.
Copyright (C) 1996-1997, Cyberlogic Technologies, Inc.
MBXAPI Demo/Diagnostics Program Version 1.00

 [1]    Set device number:   [0 dec]
        (Remote MBX Client)

 [2]    Read selected node
 [3]    Write selected node
 [4]    Read slave data path messages
 [5]    Write global data
 [6]    Read global data
 [7]    Read device status
 [8]    Device information
 [ESC]  Exit

Enter selection >_
```

2. This screen shows configuration, statistical, and diagnostic information about the driver, the device, and the network from the server's view. Hit ***<Esc>*** to get back to the main menu.



```
MbxDemo
Device Type:                SA85           Network Protocol:          MB+
Adapter Number:             0              Baud Rate:                 1.00
Memory Address:             C8000          Port Address:              0
Interrupt IRQ:              Polled Mode    Bus Type:                  ISA
Polling Interval:           20 msec        Bus Number:                0
Max Nodes:                  64             Slot Number:               N/A
Node Address:               18

Adapter Card Status:        On-Line        Total Dev Driver Calls:    6611
Device Open Count:          3              Total Error Calls:         0
DM Open/Active Count:       1/0            PM Open/Active Count:      0/0
Total DM Cmd Packets:       2243009        Total PM Cmd Packets:      1684
Total DM Reply Pkt's:       2242379        Total PM Reply Pkt's:      1684
Total DM Cmd Timeouts:      0              Total PM Cmd Timeouts:     0
DS Open/Active Count:       0/0            PS Open/Active Count:      0/0
Total DS Cmd Packets:       0              Total PS Cmd Packets:      0
Total DS Reply Pkt's:       0              Total PS Reply Pkt's:      0
Total Lost DS Cmd's:        0              Total Lost PS Cmd's:       0
Active Glb Data Reads:      0              Active Glb Data Writes:    0
Total Glb Data Reads:       591            Total Glb Data Writes:     0
Pending Adapter Req's:      0              Active Dev Stat Req's:     0
Total/Lost Interrupts:      0/0            Total Dev Stat Req's:      2
Total XMT Packets:          2244692        Total Adapter Faults:      0
Total RCV Packets:          2244692        Last Crash code:           0x0
```

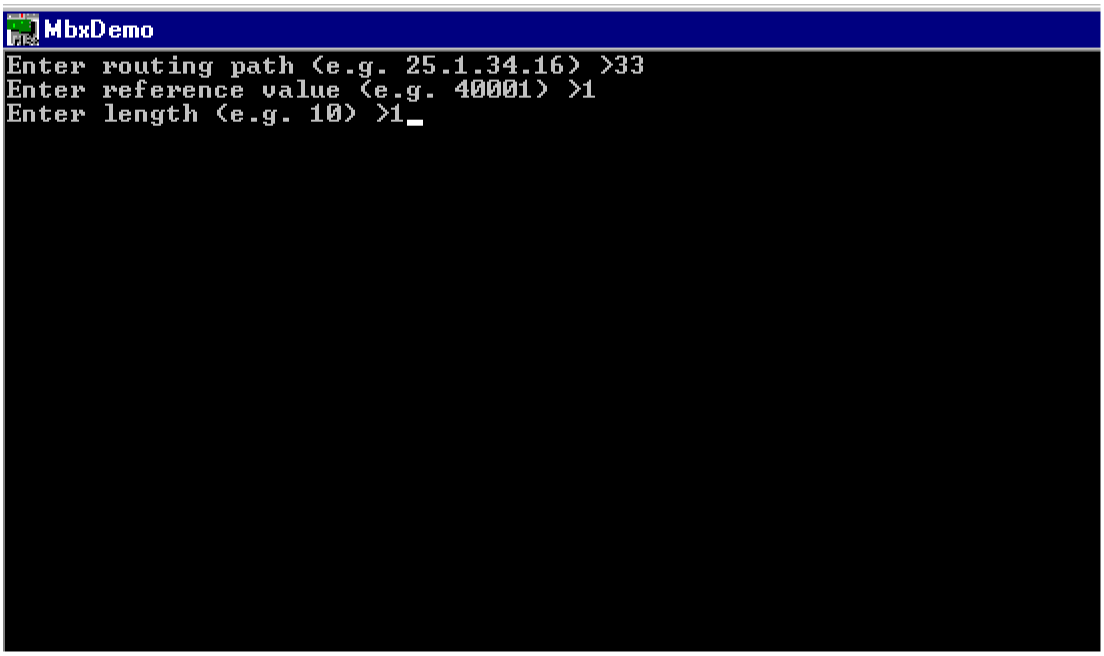
3. Select number ***[7] Read device status***. The following screen will appear. This screen should show all active nodes on the network.



```
--- Modbus Plus Network Statistics ---

Node type ID:                   Host computer
Peer processor version:         1.01
This node address:              18
MAC state:                      Check-pass
Peer status:                    32
Token pass counter:             31835
Token rotation time:            1

      ---- Active Node List ----

  1 ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦U¦ ¦ ¦ ¦ ¦
 17 ¦ ¦I¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦U¦
 33 ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦
 49 ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦
```

4. Hit ***<Esc>*** to get back to the main menu.

5. Select number ***[2] Read selected node***. You should be able to read data from any active node on the network.



MbxDemo

```
Enter routing path (e.g. 25.1.34.16) >33
Enter reference value (e.g. 40001) >1
Enter length (e.g. 10) >1_
```

4.5 Frequently Asked Questions

- I’ve installed the software. What’s next?

  The next step is to configure a device. See the *Remote MBX Driver Configuration* on page 59 for more details.

- I have two devices configured in the system. How do I communicate through the second one?

  MbxDemo uses the device number to determine which card to use. Option 1, *Set device number,* lets you choose which device the demo will use. If you are using some other software, contact the manufacturer for more information on using multiple devices.

- I have to communicate to two server nodes, each located in a separate physical subnet. Both subnets are connected through a bridge. Can I communicate to both nodes?

  As part of the device configuration, the user specifies the name of the locator node. The locator can only locate server nodes that are part of the same physical subnet. Since only one locator node can be specified per client system, only one of the server nodes will be accessible (the one with the locator node).

The 32-bit Modbus Plus Software Development Kit (SDK)

A

A.1 Overview

The Modbus Plus Software Development Kit (separately available from Schneider Automation, Part #SW-LNET-SDK) contains the 32-bit NETLIB Library. This provides an excellent bridge for developers who want to port their 16-bit applications to Windows 95 and Windows NT. All you have to do is recompile your application with a 32-bit C compiler and link it with the 32-bit NETLIB.LIB library. (Use the NETLIB.H and NETBIOS.H include files provided in the ..\RemoteNetlibLibrary directory, instead of the Schneider Automation equivalent files.)

A.2 Linking Existing Applications

To demonstrate the above capability, we have recompiled and linked Schneider Automation’s original 16-bit test programs with the 32-bit NETLIB library and provided them in the ..\RemoteNetlibLibrary directory. (Schneider Automation’s test programs are distributed with Schneider Automation’s DOS/Windows/OS/2 versions of their SA85 drivers). No changes have been made to these programs other than some minor variable declaration and include file adjustments to satisfy the Microsoft 32-bit compiler.

A.3 Developing New Applications

The 32-bit NETLIB library can also be used for new applications that are targeted for Windows 95 and Windows NT. The library is implemented to comply with and take full advantage of the multi-threaded features of Windows 95 and NT.

A.4 NETLIB Library Functions

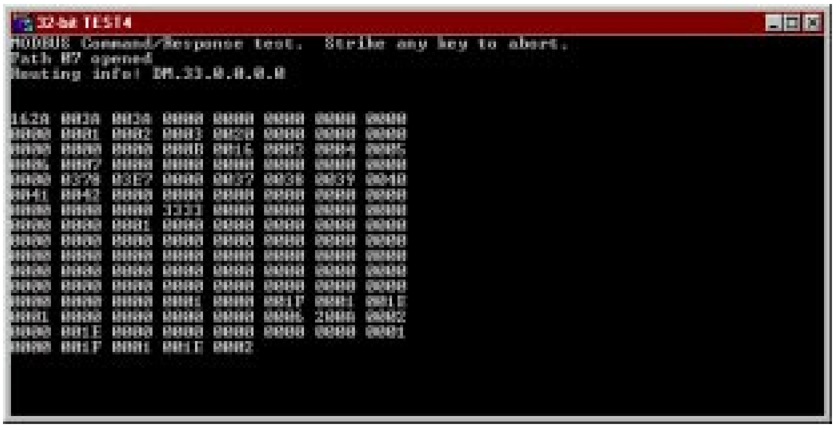
For a complete reference of all NETLIB Library functions, refer to Modicon IBM Host Based Devices User's Guide (Part# 890 USE 102 00).

A.5 System Requirements

Microsoft's Visual C++ (Version 4.2 or greater), or Borland C++ (Version 5.1 or greater), must be used when compiling. The Software Development Kit Version 4.0 is compatible with MBX drivers Version 4.0.

A.6 Software Development Kit

The following section uses Schneider Automation’s TEST4.C demo program to show the minimal amount of effort required to convert from the original 16-bit to a 32-bit version that will run with our Local NETLIB Library.



```
MODBUS Command/Response test.  Strike any key to abort.
Path 07 opened
Routing info: DM.33.0.0.0.0


162A 003A 003A 0000 0000 0000 0000 0000
0000 0001 0002 0003 0020 0000 0000 0000
0000 0000 0000 000B 0016 0003 0004 0005
0006 0007 0000 0000 0000 0000 0000 0000
0000 0378 03E7 0000 0037 0038 0039 0040
0041 0042 0000 0000 0000 0000 0000 0000
0000 0000 0000 3333 0000 0000 0000 0000
0000 0000 0001 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0001 0000 001F 0001 001E
0001 0000 0000 0000 0000 0006 2000 0002
0000 001E 0000 0000 0000 0000 0000 0001
0000 001F 0001 001E 0002
```

A.6.1 Include Files

As part of the SDK we provide two include files, NETBIOS.H and NETLIB.H, that should be used in place of the 16-bit versions of these files.

NETBIOS.H

```
/*===========================================================================
*            Copyright (C) 1994, CYBERLOGIC Technologies Inc.
*===========================================================================
*
* Module Name :
*      netbios.h
*
* Abstract:
*      This file should be included by every module that makes calls to
*      functions in the netlib.dll interface library file.
*
*
* ---------+-----+-----------------------------------------------------------
*   DATE   | BY  |  DESCRIPTION / REASON FOR MODIFICATION
* ---------+-----+-----------------------------------------------------------
* 03-09-94 | PM  |  Start of development
* ---------+-----+-----------------------------------------------------------
* ..-..-94 |     |
*=========+=====+=======================================================*/
#define NETBIOS_INCLUDED
//
// Structure of Network Control Block (NCB)
//
#ifndef NCB_INCLUDED
#define NCB_INCLUDED
#ifdef __cplusplus
extern "C" {
#endif
typedef struct _NCB
{
   UCHAR   NCB_COMMAND;          // Command
   UCHAR   NCB_RETCODE;          // Function return code
   UCHAR   NCB_LSN;              // Local session number
   UCHAR   NCB_NUM;              // Number of network name
   PUCHAR  NCB_BUFFER;           // Far pointer to message buffer
```

```
    USHORT  NCB_LENGTH;            // Length of message buffer
    UCHAR   NCB_CALLNAME[16];      // Name of session user is talking to
    UCHAR   NCB_NAME[16];          // User’s network name
    UCHAR   NCB_RTO;               // Receive time-out in 500 ms. incrs.
    UCHAR   NCB_STO;               // Send time-out - 500 ms. increments
    void  (*NCB_POST_ADDRESS) (struct _NCB_ *);// Address of "no-wait" interrupt call
    UCHAR   NCB_LANA_NUM;          // Adapter number (must be 0 or 1)
    UCHAR   NCB_CMD_CPLT;          // Command completion status
    UCHAR   NCB_RESERVE[14];       // Reserved area for Token-Ring
} NCB,*PNCB;
#else
//
// _NCB structure is already defined in nb30.h file.
// Microsoft uses lower case while Schneider Automation uses upper case letters
// for all names in this structure. The caller can use either convention.
//
#define NCB_COMMAND        ncb_command
#define NCB_RETCODE        ncb_retcode
#define NCB_LSN            ncb_lsn
#define NCB_NUM            ncb_num
#define NCB_BUFFER         ncb_buffer
#define NCB_LENGTH         ncb_length
#define NCB_CALLNAME       ncb_callname
#define NCB_NAME           ncb_name
#define NCB_RTO            ncb_rto
#define NCB_STO            ncb_sto
#define NCB_POST_ADDRESS   ncb_post
#define NCB_LANA_NUM       ncb_lana_num
#define NCB_CMD_CPLT       ncb_cmd_cplt
#define NCB_RESERVE        ncb_reserve
#endif
//
// NetBIOS functions list - "WAIT" calls wait until command completes
// while the others jump to the routine in NCB_POST when the NetBIOS
// command completes and does an interrupt.
//
#define RESET              0x32    // Reset adapter card and tables
#define CANCEL             0x35    // Cancel command. NCB_BUFFER = cmd.
#define STATUS             0xb3    // Status information for adapter
```

```
#define STATUS_WAIT              0x33   //
#define TRACE                    0xf9   // Token-Ring protocol trace
#define TRACE_WAIT               0x79   //
#define UNLINK                   0x70   // Unlink from IBM Remote Program
#define ADD_NAME                 0xb0   // Add name to name table
#define ADD_NAME_WAIT            0x30   //
#define ADD_GROUP_NAME           0xb6   // Add group name to name table
#define ADD_GROUP_NAME_WAIT      0x36   //
#define DELETE_NAME              0xb1   // Delete name from name table
#define DELETE_NAME_WAIT         0x31   //
#define CALL                     0x90   // Start session with NCB_NAME name
#define CALL_WAIT                0x10   //
#define LISTEN                   0x91   // Listen for call
#define LISTEN_WAIT              0x11   //
#define HANG_UP                  0x92   // End session with NCB_NAME name
#define HANG_UP_WAIT             0x12   //
#define SEND                     0x94   // Send data via NCB_LSN
#define SEND_WAIT                0x14   //
#define SEND_NO_ACK              0xf1   // Send data without waiting for ACK
#define SEND_NO_ACK_WAIT         0x71   //
#define CHAIN_SEND               0x97   // Send multiple data buffers
#define CHAIN_SEND_WAIT          0x17   //
#define CHAIN_SEND_NO_ACK        0xf2   // Send multiple buffers without ACK
#define CHAIN_SEND_NO_ACK_WAIT   0x72   //
#define RECEIVE                  0x95   // Receive data from a session
#define RECEIVE_WAIT             0x15   //
#define RECEIVE_ANY              0x96   // Receive data from any session
#define RECEIVE_ANY_WAIT         0x16   //
#define SESSION_STATUS           0xb4   // Status of all sessions for name
#define SESSION_STATUS_WAIT      0x34   //
#define SEND_DATAGRAM            0xa0   // Send un-ACKed message
#define SEND_DATAGRAM_WAIT       0x20   //
#define SEND_BCST_DATAGRAM       0xa2   // Send broadcast message
#define SEND_BCST_DATAGRAM_WAIT  0x22   //
#define RECEIVE_DATAGRAM         0xa1   // Receive un-ACKed message
#define RECEIVE_DATAGRAM_WAIT    0x21   //
#define RECEIVE_BCST_DATAGRAM    0xa3   // Receive broadcast message
#define RECEIVE_BCST_DATAGRAM_WAIT 0x23   //
#define SA85_OFF                 0x37   // Turn SA85 driver off (RESET to turn on)
```

```
#define SA85_MASK             0x38    // Return status of mask bit
#define SA85_ION              0x39    // Enable sa85 interrupts
#define SA85_IOFF             0x3a    // Disable sa85 interrupts
#define SET_SLAVE_LOGIN       0x3b    // Set/clear slave login status
//
// NetBIOS error return codes - returned in NCB_RETCODE
//
#ifndef ERR_success
#define ERR_success           0       // NetBIOS command completed normally
#define ERR_bad_buffer_length 1       // Bad send or status buffer size
#define ERR_invalid           3       // invalid NetBIOS command
#define ERR_timeout           5       // Command time-out has expired
#define ERR_buffer_too_small  6       // Receive buffer not big enough
#define ERR_bad_session_num   8       // Bad value in NCB_LSN
#define ERR_no_RAM            9       // LAN card doesn’t have enough memory
#define ERR_session_closed    0xa     // This session is closed
#define ERR_cancel            0xb     // Command has been closed
#define ERR_dup_local_name    0xd     // Name already exists for this PC
#define ERR_name_table_full   0xe     // Local name table is full
#define ERR_active_session    0xf     // Can’t delete name - used in session
#define ERR_sess_table_full   0x11    // Local session table is full
#define ERR_no_listen         0x12    // Remote PC not listening for call
#define ERR_bad_name_num      0x13    // Bad value in NCB_NUM field
#define ERR_no_answer         0x14    // No answer to CALL or no such remote
#define ERR_no_local_name     0x15    // No such name in local name table
#define ERR_duplicate_name    0x16    // Name is in use elsewhere on net
#define ERR_bad_delete        0x17    // Name incorrectly deleted
#define ERR_abnormal_end      0x18    // Session aborted abnormally
#define ERR_name_error        0x19    // 2 or more identical names in use!
#define ERR_bad_packet        0x1a    // Bad NetBIOS packet on network
#define ERR_card_busy         0x21    // network card is busy
#define ERR_too_many_cmds     0x22    // Too many NetBIOS commands queued
#define ERR_bad_card_num      0x23    // bad NCB_LANA_NUM - must be 0 or 1
#define ERR_cancel_done       0x24    // command finished while cancelling
#define ERR_no_cancel         0x26    // Command can’t be cancelled
#define ERR_busy              0xff    // Still processing command
#ifdef __cplusplus
}
#endif
```

```
#endif // NCB_INCLUDED
```

NETLIB.H

```
/*=====================================================================
*           Copyright (C) 1994, CYBERLOGIC Technologies Inc.
*=====================================================================
*
* Module Name :
*       netlib.h
*
* Abstract:
*       This file should be included by every module that makes calls to
*       functions in the netlib.dll interface library file.
*
*
* ---------+-----+----------------------------------------------------------
*   DATE   | BY  |  DESCRIPTION / REASON FOR MODIFICATION
* ---------+-----+----------------------------------------------------------
* 03-09-94 | PM  |  Start of development
* ---------+-----+----------------------------------------------------------
* ..-..-94 |     |
*=========+=====+====================================================*/
#ifndef _NETLIB_H_
#define _NETLIB_H_
//
// Define API decoration for direct importing of DLL references.
//
#ifdef _DLLEXPORT_NETLIB_
#define NETLIBAPI
#else
#define NETLIBAPI DECLSPEC_IMPORT
#endif // _DLLEXPORT_NETLIB_
#ifdef __cplusplus
extern "C" {
#endif
/*==================================================================*/
/* Include files
/*==================================================================*/
#ifndef NETBIOS_INCLUDED
#include "netbios.h"
#endif
/*==================================================================*/
```

```
/* Function prototypes
/*==================================================================*/
NETLIBAPI
int
APIENTRY
ncb_reset(
   int adaptno
   );
NETLIBAPI
int
APIENTRY
ncb_sa85off(
   int adaptno
   );
NETLIBAPI
int
APIENTRY
ncb_status(
   PNCB ncbp,
   int adaptno
   );
NETLIBAPI
int
APIENTRY
ncb_send(
   PNCB ncbp,
   int length,
   PCHAR buffer,
   UCHAR timeout
   );
NETLIBAPI
int
APIENTRY
ncb_receive_wait(
   PNCB ncbp,
   PCHAR buffer,
   UCHAR timeout
   );
NETLIBAPI
```

```
PNCB
APIENTRY
ncb_open(
   PCHAR name,
   int lan
   );
NETLIBAPI
int
APIENTRY
ncb_receive(
   PNCB ncbp,
   PCHAR buffer
   );
NETLIBAPI
int
APIENTRY
ncb_close(
   PNCB ncbp
   );
NETLIBAPI
int
APIENTRY
ncb_send_datagram(
   PNCB ncbp,
   int length,
   PCHAR buffer,
   UCHAR timeout,
   int adaptno
   );
NETLIBAPI
int
APIENTRY
ncb_receive_datagram(
   PNCB ncbp,
   int node,
   PCHAR buffer,
   UCHAR timeout,
   int adaptno
   );
```

```
NETLIBAPI
int
APIENTRY
ncb_cancel(
   PNCB ncbp
   );
NETLIBAPI
int
APIENTRY
ncb_set_slave_login(
   PNCB ncbp,
   UCHAR login_status
   );
NETLIBAPI
int
APIENTRY
ncb_set_sw_interrupt(
   int swInterrupt
   );
#ifdef __cplusplus
}
#endif
#endif //_NETLIB_H_
```

A.6.2 Source Files

The following shows both the 16-bit as well as the 32-bit source code for the TEST4.C programs. We highlighted the differences between the two versions to show the minimal effort required to do the conversion.

16 Bit Version of TEST4.C

```
/*name test4.c*/
/* Copyright (C) Schneider Automation, Inc. 1989, All Rights Reserved. */
/*
This test program uses the SA85.SYS device driver to read 125 holding
registers from the MODBUS slave which is found at the given node number.
In order to use this program, enter the following command:
   A>TEST4 12
where "12" is the slave node we want to read from.
*/
/*
```

```
include
*/
#include <stdio.h>
#include <dos.h>
#include <process.h>
#include <string.h>
#include <conio.h>
#include <signal.h>
#include "netbios.h"
#include "netlib.h"
/*
prototypes
*/
int main(int argc, char *argv[] );
void dump(int qty, unsigned int *buff);
void control_c(void);
/*
global variables
*/
char mbuffer[256];
char path[ 80 ];
int completed;
/*
main()
*/
int main( argc, argv )
int argc;
char *argv[];
{
   NCB *nd;
   int ret_val;
   int i;
   printf( "MODBUS Command/Response test. Strike any key to abort." );
   if( argc < 2 ) {
      printf( "Usage is: A>TEST4 <slave node>");
      exit( 1 );
   }
   ret_val = sscanf( argv[ 1 ], "%d", &i );
   if( ret_val != 1 || i < 1 || i > 64 ) {
    printf( "Node number must be in the range 1 to 64. " );
    exit( 1 );
}
if( signal( SIGINT, control_c ) == SIG_ERR ) {
    printf( "Unable to redirect Control-C." );
    exit( 1 );
}

sprintf( path, "DM.%d.0.0.0.0", i );
if ((nd = ncb_open( path, 0 )) == NULL) {
    printf("Unable to open DATA MASTER path.");
    exit(1);
    }
printf("Path %02X opened", nd->NCB_NUM);
printf("Routing info: %c%c.%d.%d.%d.%d.%d",
    nd->NCB_CALLNAME[0],
    nd->NCB_CALLNAME[1],
    nd->NCB_CALLNAME[2],
    nd->NCB_CALLNAME[3],
    nd->NCB_CALLNAME[4],
    nd->NCB_CALLNAME[5],
    nd->NCB_CALLNAME[6]);
completed = 0;          /* global variable, do while zero */
while( !completed ) {
                  if( kbhit() )
        completed = 1;
    /*
     * Note: filling in mbuffer[0] with the slave address
     * is actually unecessary, since it was specified above
     * in the ncb_open. It is done here, and the buffer
     * pointer adjusted by 1 in ncb_send, to provide ease-of-use
     * and compatibility with existing modbus applications.
   */
    mbuffer[0] = 0x5;   /*slave address*/
    mbuffer[1] = 0x3;   /*command*/
    mbuffer[2] = 0x0;   /*offset high*/
    mbuffer[3] = 0x0;   /*offset low*/
    mbuffer[4] = 0x0;   /*reg count high*/
    mbuffer[5] = 125;   /* reg count low*/
```

```
        if (ncb_send(nd, 6, mbuffer, 10) != 0) {
                    /*send the command*/
          printf("Send error: %d.", nd->NCB_RETCODE);
          ret_val = ncb_close(nd);    /*close the path*/
          exit(1);
          }
        if (ncb_receive_wait(nd, mbuffer, 10) != 0)
                    /*try to receive*/
          printf("Receive error: %d.", nd->NCB_RETCODE);
        else
                            dump( 125, (unsigned int *) &mbuffer[ 3 ]);
      }
      ncb_close(nd);      /*close the path*/
      exit(0);            /* good exit */
      return( 0 );
    }
    /*
    dump
    Dump the contents of the buffer for the length specified.
    */
    void dump(qty, buff)
    int qty;                                    /* qty of bytes to dump */
    unsigned int *buff;       /* buffer with the data */
             {                                              /* dump_lines */
             int i = 0, j;
      unsigned int out_value;
             printf("");
             do        {
                       printf("");
                       for(j = 0;  j < 8 && i < qty;  j++,  i++) {
          out_value = ((*buff << 8) & 0xff00) | ((*buff >> 8) & 0xff);
          printf("%04X ", out_value);
          buff++;
        }
      } while (i < qty);
             printf("");
             }                                              /* dump_lines */
    /*
    control_c
```

```
This routine replaces the control-C handler supplied by DOS. When you
type a control_C, the program will vector to this routine, which will
set the completed flag to non-zero. In this test program, this will cause
the infinite loop within main to terminate.
*/
void control_c()
{
    signal( SIGINT, SIG_IGN );          /* disable control-c */
    completed = 1;                      /* will cause main loop to complete */
    signal( SIGINT, control_c );        /* reset control-c handler */
}
```

32 Bit Version of TEST4.C

```
/*name test4.c*/
/* Copyright (C) Schneider Automation, Inc. 1989,  All Rights Reserved. */
/*
This test program uses the SA85.SYS device driver to read 125 holding
registers from the MODBUS slave which is found at the given node number.
In order to use this program, enter the following command:
    A>TEST4 12
where "12" is the slave node we want to read from.
*/
/*
include
*/
#define STRICT
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <signal.h>
#include "netbios.h"
#include "netlib.h"
/*
prototypes
*/
int main(int argc, char *argv[] );
void dump(int qty, PUSHORT buff);
void control_c(int sig);
/*
```

```
global variables
*/
char mbuffer[256];
char path[ 80 ];
int completed;
/*
main()
*/
int main( argc, argv )
int argc;
char *argv[];
{
   NCB *nd;
   int ret_val;
   int i;
   printf( "MODBUS Command/Response test. Strike any key to abort." );
   if( argc < 2 ) {
      printf( "Usage is: A>TEST4 <slave node>");
      exit( 1 );
   }
   ret_val = sscanf( argv[ 1 ], "%d", &i );
   if( ret_val != 1 || i < 1 || i > 64 ) {
      printf( "Node number must be in the range 1 to 64. " );
      exit( 1 );
   }
   if( signal( SIGINT, control_c ) == SIG_ERR ) {
      printf( "Unable to redirect Control-C." );
      exit( 1 );
   }

   sprintf( path, "DM.%d.0.0.0.0", i );
   if ((nd = ncb_open( path, 0 )) == NULL) {
      printf("Unable to open DATA MASTER path.");
      exit(1);
      }
   printf("Path %02X opened", nd->NCB_NUM);
   printf("Routing info: %c%c.%d.%d.%d.%d.%d",
      nd->NCB_CALLNAME[0],
      nd->NCB_CALLNAME[1],
      nd->NCB_CALLNAME[2],
      nd->NCB_CALLNAME[3],
      nd->NCB_CALLNAME[4],
      nd->NCB_CALLNAME[5],
      nd->NCB_CALLNAME[6]);
   completed = 0;          /* global variable, do while zero */
   while( !completed ) {
                   if( kbhit() )
           completed = 1;
      /*
       * Note: filling in mbuffer[0] with the slave address
       * is actually unecessary, since it was specified above
       * in the ncb_open. It is done here, and the buffer
       * pointer adjusted by 1 in ncb_send, to provide ease-of-use
       * and compatibility with existing modbus applications.
      */
      mbuffer[0] = 0x5;   /*slave address*/
      mbuffer[1] = 0x3;   /*command*/
      mbuffer[2] = 0x0;   /*offset high*/
      mbuffer[3] = 0x0;   /*offset low*/
      mbuffer[4] = 0x0;   /*reg count high*/
      mbuffer[5] = 125;   /* reg count low*/
      if (ncb_send(nd, 6, mbuffer, 10) != 0) {
                   /*send the command*/
           printf("Send error: %d.", nd->NCB_RETCODE);
           ret_val = ncb_close(nd);    /*close the path*/
           exit(1);
           }
      if (ncb_receive_wait(nd, mbuffer, 10) != 0)
                   /*try to receive*/
           printf("Receive error: %d.", nd->NCB_RETCODE);
      else
                              dump( 125, (PUSHORT) &mbuffer[ 3 ]);
   }
   ncb_close(nd);      /*close the path*/
   exit(0);            /* good exit */
   return( 0 );
}
/*
```

```
dump
Dump the contents of the buffer for the length specified.
*/
void dump(qty, buff)
int qty;                                    /* qty of bytes to dump */
PUSHORT buff;           /* buffer with the data */
          {                                                /* dump_lines */
          int i = 0, j;
  unsigned int out_value;
          printf("");
          do          {
                    printf("");
                    for(j = 0;  j < 8 && i < qty;  j++,  i++) {
      out_value = ((*buff << 8) & 0xff00) | ((*buff >> 8) & 0xff);
      printf("%04X ", out_value);
      buff++;
    }
  } while (i < qty);
          printf("");
          }                                                /* dump_lines */
/*
control_c
This routine replaces the control-C handler supplied by DOS. When you
type a control_C, the program will vector to this routine, which will
set the completed flag to non-zero. In this test program, this will cause
the infinite loop within main to terminate.
*/
void control_c(int sig)
{
  signal( SIGINT, SIG_IGN );       /* disable control-c */
  completed = 1;                /* will cause main loop to complete */
  signal( SIGINT, control_c );     /* reset control-c handler */
}
```

A.6.3 Makefile

The following makefile is used to build all of the demo programs. It has been tested with the 32-bit version of Microsoft Visual C++.

```
#
#     +==========================================================+
#     |     Copyright (C) 1995, CYBERLOGIC Technologies, Inc.     |
#     +==========================================================+
#
#
# Nmake macros for building Windows 32-Bit apps
#
# To run production release enter:
#   >nmake nodebug=1
#
# To remove all but the final target files enter:
#   >nmake clean
#
# To remove all target files enter:
#   >nmake cleanall
#
TARGETOS=WIN95
!include <ntwin32.mak>
!IFDEF NODEBUG
!    IF $(NODEBUG) == 1
BLDENV = free
!    ELSE
BLDENV = checked
!    ENDIF
!ELSE
BLDENV = checked
!ENDIF
!IF "$(CYTARGETPATH)" == ""
BLDTARGET = .
!ELSE
BLDTARGET = $(CYTARGETPATH)(BLDENV)
!ENDIF
BLDOBJ = obj
!IF "$(CYLIBPATH)" == ""
BLDLIB = .
!ELSE
BLDLIB = $(CYLIBPATH)(BLDENV)
!ENDIF
```

```
!IF "$(CYINCPATH)" == ""
BLDINC = .
!ELSE
BLDINC = $(CYINCPATH)
!ENDIF
all:    $(BLDTARGET)est4.exe
        $(BLDTARGET)est4b.exe
        $(BLDTARGET)est4c.exe
        $(BLDTARGET)estslav.exe
        $(BLDTARGET)lobtest.exe
        $(BLDTARGET)eadnode.exe
$(BLDOBJ)est4.obj: test4.c $(BLDINC)etlib.h $(BLDINC)etbios.h
    $(cc) $(cdebug) $(cflags) $(cvarsdll) -I$(BLDINC)
        /Fo$(BLDOBJ)est4.obj test4.c
$(BLDOBJ)est4b.obj: test4b.c $(BLDINC)etlib.h $(BLDINC)etbios.h
    $(cc) $(cdebug) $(cflags) $(cvarsdll) -I$(BLDINC)
        /Fo$(BLDOBJ)est4b.obj test4b.c
$(BLDOBJ)est4c.obj: test4c.c $(BLDINC)etlib.h $(BLDINC)etbios.h
    $(cc) $(cdebug) $(cflags) $(cvarsdll) -I$(BLDINC)
        /Fo$(BLDOBJ)est4c.obj test4c.c
$(BLDOBJ)estslav.obj: testslav.c $(BLDINC)etlib.h $(BLDINC)etbios.h
    $(cc) $(cdebug) $(cflags) $(cvarsdll) -I$(BLDINC)
        /Fo$(BLDOBJ)estslav.obj testslav.c
$(BLDOBJ)lobtest.obj: globtest.c $(BLDINC)etlib.h $(BLDINC)etbios.h
    $(cc) $(cdebug) $(cflags) $(cvarsdll) -I$(BLDINC)
        /Fo$(BLDOBJ)lobtest.obj globtest.c
$(BLDOBJ)eadnode.obj: readnode.c $(BLDINC)etlib.h $(BLDINC)etbios.h
    $(cc) $(cdebug) $(cflags) $(cvarsdll) -I$(BLDINC)
        /Fo$(BLDOBJ)eadnode.obj readnode.c
$(BLDTARGET)est4.exe: $(BLDOBJ)est4.obj $(BLDLIB)etlib.lib
    $(link) $(linkdebug) $(conflags) -out:$(BLDTARGET)est4.exe
        $(BLDOBJ)est4.obj $(BLDLIB)etlib.lib $(guilibsdll)
$(BLDTARGET)est4b.exe: $(BLDOBJ)est4b.obj $(BLDLIB)etlib.lib
    $(link) $(linkdebug) $(conflags) -out:$(BLDTARGET)est4b.exe
        $(BLDOBJ)est4b.obj $(BLDLIB)etlib.lib $(guilibsdll)
$(BLDTARGET)est4c.exe: $(BLDOBJ)est4c.obj $(BLDLIB)etlib.lib
    $(link) $(linkdebug) $(conflags) -out:$(BLDTARGET)est4c.exe
        $(BLDOBJ)est4c.obj $(BLDLIB)etlib.lib $(guilibsdll)
$(BLDTARGET)estslav.exe: $(BLDOBJ)estslav.obj $(BLDLIB)etlib.lib
```

```
    $(link) $(linkdebug) $(conflags) -out:$(BLDTARGET)estslav.exe
        $(BLDOBJ)estslav.obj $(BLDLIB)etlib.lib $(guilibsdll)
$(BLDTARGET)lobtest.exe: $(BLDOBJ)lobtest.obj $(BLDLIB)etlib.lib
    $(link) $(linkdebug) $(conflags) -out:$(BLDTARGET)lobtest.exe
        $(BLDOBJ)lobtest.obj $(BLDLIB)etlib.lib $(guilibsdll)
$(BLDTARGET)eadnode.exe: $(BLDOBJ)eadnode.obj $(BLDLIB)etlib.lib
    $(link) $(linkdebug) $(conflags) -out:$(BLDTARGET)eadnode.exe
        $(BLDOBJ)eadnode.obj $(BLDLIB)etlib.lib $(guilibsdll)
# Clean all
cleanall: clean
    -del $(BLDTARGET)est4.exe
    -del $(BLDTARGET)est4b.exe
    -del $(BLDTARGET)est4c.exe
    -del $(BLDTARGET)estslav.exe
    -del $(BLDTARGET)lobtest.exe
    -del $(BLDTARGET)eadnode.exe
# Clean all but .DLL, .LIB and .EXE
clean:
    -del $(BLDOBJ).obj
    -del $(BLDTARGET)est4.map
    -del $(BLDTARGET)est4.pdb
    -del $(BLDTARGET)est4b.map
    -del $(BLDTARGET)est4b.pdb
    -del $(BLDTARGET)est4c.map
    -del $(BLDTARGET)est4c.pdb
    -del $(BLDTARGET)estslav.map
    -del $(BLDTARGET)estslav.pdb
    -del $(BLDTARGET)lobtest.map
    -del $(BLDTARGET)lobtest.pdb
    -del $(BLDTARGET)eadnode.map
    -del $(BLDTARGET)eadnode.pdb
```

Modicon, Square D and Telemecanique are PLC brand names from Schneider. These products are sold in the US by Square D; in Canada, Latin America, Europe, Africa, Asia/Pacific and Middle East by Schneider; in Germany by AEG Schneider Automation; in China and Persian Gulf by Schneider Automation; in South Africa by ASA Systems Automation; in Austria by Online.

Schneider Automation, Inc.	Schneider Automation GmbH	Schneider Automation S.A.
One High Street	Steinheimer Strasse 117	245, Route des Lucioles-BP147
North Andover, MA 01845	D-63500 Seligenstadt	F-06903 Sophia-Antipolis Cedex
Tel: (1) 508-794-0800	Tel: (49) 6182 81-2584	Tel: (33) 92 96 20 00
Fax: (1) 508-975-9400	Fax: (49) 6182 81-2860	Fax: (33) 93 65 37 15