# Modicon

# Modbus Plus Network

# IBM Host Based Devices

# User's Guide

890 USE 102 00

09/98

GROUPE SCHNEIDER

■ Modicon ■ Square D ■ Telemecanique

XXXXXX.00

**Breite: 185 mm**
**Höhe: 230 mm**

**Breite: 178 mm**
**Höhe: 216 mm**

# Preface

### Data, Illustrations, Alterations

Data and illustrations are not binding. We reserve the right to alter products in line with our policy of continuous product development. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us using the form on one of the last pages of this publication.

### Training

Schneider Automation offers suitable further training on the system.

### Trademarks

All terms used in this publication to denote Schneider Automation products are trademarks of Schneider Automation Incorporated.

All other terms used in this publication to denote products may be registered trademarks and/or trademarks of the corresponding corporations.

Microsoft and MS-DOS are registered trademarks of Microsoft Corporation, Windows is a brandname of Microsoft Corporation in the USA and other countries.

IBM is a registered trademark of International Business Machines Corporation.

Intel is a registered trademark of Intel Corporation.

### Copyright

# Contents

## Chapter 1
## Introducing the Host Based Devices

## Chapter 2
## The AT-984 Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

## Chapter 3
## The SA85 Network Adapter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

## Chapter 4
## The MC-984 Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

**Breite: 185 mm**

## Chapter 6
## Installing the DOS Files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85

## Chapter 7
## Installing the OS/2 Files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89

## Appendix A.
## Using NetBIOS Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93

## Appendix B.
## Modifying the 5C Interrupt . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125

## Appendix C. Include Files . . . . . 129

## Appendix D. The MBPSTAT Utility . . . . . 137

## Appendix F.
## Using Modbus Commands . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163

## Appendix G.
## Modbus Plus Statistics Explained . . . . . . . . . . . . . . . . . . . . . . . . . . 193

## Appendix H.
## Updating Your Product . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 209

## Illustrations

# Chapter 1
# Introducing the Host Based Devices

□ Modicon Host Based Devices

□ An Application Overview

□ The SA85 and SM85 Adapters

□ The AT-984 and MC-984 Controllers

□ Modbus Plus Message Routing

□ Modbus Plus Transactions

□ Paths in Host Based Controllers

□ Modbus Data Access Commands

□ More Programming Information

## 1.1 Modicon Host Based Devices

This book describes the installation, setup, and programming of four adapters for Modbus Plus networking in host computers using IBM AT or Micro Channel bus architecture. The four adapters are:

□ The SA85 Modbus Plus Network Adapter (part numbers AM- SA85- 000, -002), which establishes an IBM- AT or compatible host as a Modbus Plus node

□ The SM85 Modbus Plus Network Adapter (part number AM-SM85-000), which establishes an IBM PS/2 host as a Modbus Plus node

□ The AT-984 Programmable Controller (part numbers AM-0984-A T0, -AT2, -AT4), which provides 984 control capability for an IBM-AT or compatible host, in addition to establishing the host as a Modbus Plus node

□ The MC-984 Programmable Controller (part number AM-0984-MC0), which provides 984 control capability on a board for an IBM PS/2 host, in addition to establishing the host as a Modbus Plus node.

The AT-984 and SA85 are to be installed in an IBM AT-compatible motherboard. The MC-984 and the SM85 are to be installed in an IBM PS/2 motherboard. Each device occupies one slot in the host computer's motherboard. Up to two Modicon devices can be installed within a single host.

### AT–Based Products for Single and Dual Cable Networks

Two kinds of AT-based versions are available. AM-SA85-000 and AM-0984-A T0 connect to a Modbus Plus network that uses a single cable between nodes. AM-SA85-002, AM-0984-AT2, and AM-0984-A T4 connect to a dual-cable network.

### Available Software

Application packages supported by the host based devices include Modicon MODSOFT programming software, Modcom III communications library software, and third party packages. Contact Modicon for information about application software for your system.

### 1.1.1 Distribution Software

The AT-based devices (SA85 and AT-984) include a set of software programs on 5.25-inch and 3.5-inch diskettes. The Micro Channel based devices (SM85 and MC-984) include software on 3.5-inch diskettes. Your diskettes contain:

□ Device drivers for DOS and OS/2 that allow you to customize the presence of each device in your host computer

□ Setup programs used by the Micro Channel devices to define the host interface

□ A library of C language functions for Modicon's NetBIOS implementation, containing a subset of IBM NetBIOS for Modbus Plus networking

□ A diagnostic utility that allows you to monitor Modbus Plus network activity and record error statistics from remote nodes

□ A set of sample programs that demonstrate methods of Modbus Plus network addressing and data transfers. In addition to the executable files, C source code is supplied to illustrate how the communications library functions can be used in your application programs.

Your diskettes have a PACKING.LST text file that lists all of the programs.

### 1.1.2 Information Text Files

Any recently updated information about the installation of your host based device is contained in a text file on your diskettes. The file names are README.DOC (DOS) and READ.ME (OS/2).

Use your operating system's TYPE command to view the file applicable to your system, or use your PRINT or COPY (to LPT1:) commands to print a hard copy. You should review the file before installing your host based device.

### 1.1.3 Source Code Files

Your diskettes contain the function library NETLIB.C, and the full source code for your set of sample programs. You can print a hard copy of these files to use as a reference in coding your application.

## 1.2 An Application Overview

Each host based device is a peer node on the Modbus Plus network, handling the network token in its assigned address sequence. Host applications can send and receive data messages, retrieve network statistics, and access the network's global database. In addition to its network functions, each 984 controller can control industrial processes through its remote input/output port, using a stored program of ladder logic instructions.

Both single-cable and dual-cable networks are supported by AT-based devices.

Figure 1 shows typical connections for the host based devices.

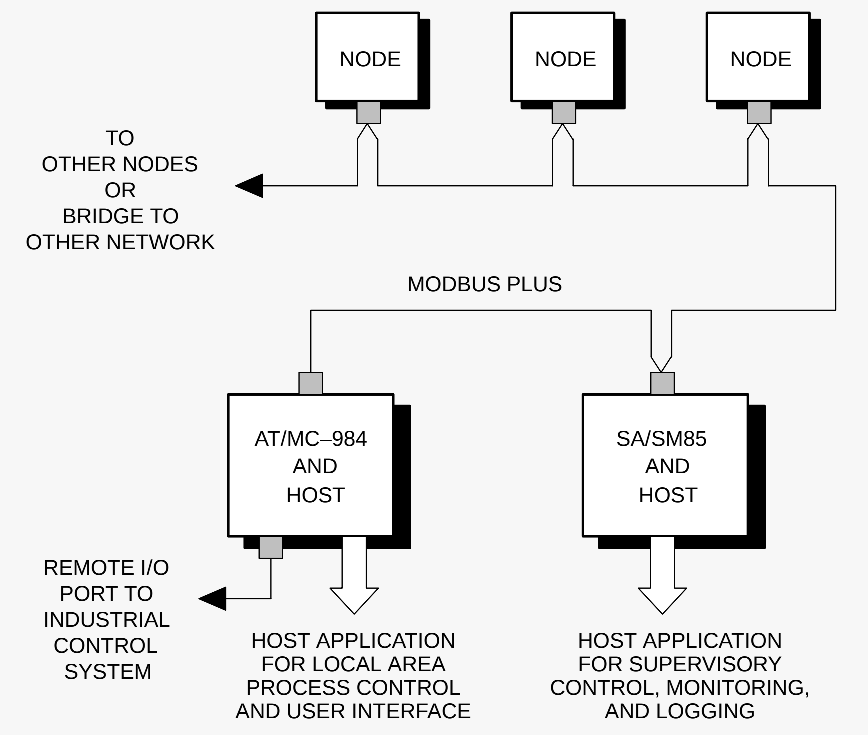

NODE
NODE
NODE
TO OTHER NODES OR BRIDGE TO OTHER NETWORK
MODBUS PLUS
AT/MC–984 AND HOST
SA/SM85 AND HOST
REMOTE I/O PORT TO INDUSTRIAL CONTROL SYSTEM
HOST APPLICATION FOR LOCAL AREA PROCESS CONTROL AND USER INTERFACE
HOST APPLICATION FOR SUPERVISORY CONTROL, MONITORING, AND LOGGING

**Figure 1 Application Overview**

## 1.3 The SA85 and SM85 Adapters

The SA85 Modbus Plus Network Adapter connects an IBM AT or compatible host to the network. It is a half-size circuit board that plugs into the host motherboard's standard 8-bit option bus. Single- and dual-cable network models are available.

The SM85 Modbus Plus Network Adapter connects an IBM PS/2 or compatible host to the network. It is a full-size circuit board that plugs into the host motherboard's Micro Channel bus.

Each adapter is provided with a C library that applies Modicon's implementation of NetBIOS commands to Modbus Plus networking. DOS and OS/2 drivers are also included to interface each adapter to the host application.

### 1.3.1 Network Adapter Applications

User applications running in the host computer can access the network adapter to read/write data at nodes on the local network, and at nodes on remote networks through bridges. The host application can also program network nodes remotely, retrieve statistics, and read/write data directly to the network's global database. Typical applications include:

□ Control, monitoring, and reporting of remote processes

□ Program load/record/verify operations

□ User interfaces

□ Bridging between Modbus Plus and other types of networks

□ Testing and debugging of application programs

□ Running network diagnostic programs.

Standard Modicon Modbus commands originated from 984 controller nodes can be addressed to a network adapter, and given to tasks running in the host. Examples include:

□ Running a data logging task in the host, accessed by 984 controllers and other nodes on the network

□ Providing virtual registers for remote 984 controllers.

# 1.4 The AT–984 and MC–984 Controllers

In addition to their Modbus Plus network functions, the AT-984 and MC-984 boards offer standard 984 control capabilities. Both boards provide the following ladder logic programming elements and logic instructions. For more details on these instructions, see the *984 Controller Systems Manual*.

**Standard Programming Elements**

| Symbol | Meaning |
|---|---|
| –\|   \|– | Normally open (N.O.) contact |
| –\| \ \|– | Normally closed (N.C.) contact |
| –\| ↑ \|– | Positive transitional (P.T.) contact |
| –\| ↓ \|– | Negative transitional (N.T.) contact |
| –(   )– | Normal coil |
| –( L )– | Latched (memory-retentive) coil |

**Instruction Set**

| Type | Instruction | Meaning |
|---|---|---|
| Counters and Timers | UCTR | Counts up from 0 to a preset value |
| | DCTR | Counts down from a preset value to 0 |
| | T1.0 | Timer that measures in seconds |
| | T0.1 | Timer that measures in tenths of a second |
| | T.01 | Timer that measures in hundredths of a second |
| Calculations | ADD | Adds top node value to middle node value |
| | SUB | Subtracts middle node value from top node value |
| | MUL | Multiplies top node value by middle node value |
| | DIV | Divides top node value by middle node value |
| | EMTH | Lets you select from a library of 38 enhanced math operations, including floating point operations and extra integer math operations such as square root |
| DX Move | R→T | Moves register values to a table |
| | T→R | Moves table values to a register |
| | T→T | Moves a specified set of values from one table to another table |
| | BLKM | Moves a specified block of data |
| | TBLK | Moves a block of data from a table to another specified block area |
| | BLKT | Moves a block of registers to specified locations in a table |
| | FIN | First–in operation to a queue |
| | FOUT | First–out operation to a queue |
| | SRCH | Performs a table search |

| Type | Instruction | Meaning |
|---|---|---|
| DX Move (continued) | READ | Reads an ASCII input device message to the PLC memory |
| | WRIT | Writes a message from the controller to an ASCII output device |
| DX Matrix | AND | Logically ANDs the contents of two matrices |
| | OR | Performs logical inclusive OR of the contents of two matrices |
| | XOR | Performs logical exclusive OR of the contents of two matrices |
| | COMP | Performs the logical complement of values in a source matrix |
| | CMPR | Logically compares the values in two matrices |
| | MBIT | Performs a logical bit modify operation |
| | SENS | Performs a logical bit sensing operation |
| | BROT | Performs a logical bit rotate operation |
| System health | STAT | Displays a group of status registers describing the health of the RIO network |
| Closed loop control | PID2 | Performs a specified proportional–integral–derivative function |
| Modbus Plus Networking | MSTR | Specifies a function from a menu of networking operations |
| Skip nodes | SKP | Skips a specified number of networks in a ladder logic program |
| Drum sequencing | SCIF | Emulates a mechanical tenor drum, letting you perform sequential control and input comparison operations |
| Subroutine/interrupt support | JSR | Jumps from scheduled logic scan to a ladder logic subroutine |
| | LAB | Labels the entry point in a ladder logic network where the subroutine or interrupt logic begins |
| | RET | Returns the logic scan from the subroutine or interrupt logic to its previous position in the logic program |

Software loadable instructions may also be added to an AT-984 or MC-984:

**Loadable Instructions**

| Instruction | Meaning |
|---|---|
| FN*xx* | Allows you to develop your own custom loadable function blocks |
| DRUM and ICMP | Simplifies implementation of sequential step oriented logic and other software loadable functions. See the manual noted above for a list of instructions applicable to this and other 984 controllers. |

## 1.5 Modbus Plus Message Routing

Each device establishes its host as a node on the Modbus Plus network. Each node is assigned a unique address between 1 and 64 on its network. Multiple networks can be joined through Bridge Plus devices.

### 1.5.1 Modbus Plus Routing Paths

Nodes address each other using a routing path field of five bytes. The path is imbedded in the Modbus Plus message frame as sent from the originating node. The five bytes of routing allow destination nodes to be addressed up to four networks away from the originating node. The routing bytes are used by each type of device in a specific way, as illustrated below and on the next page.

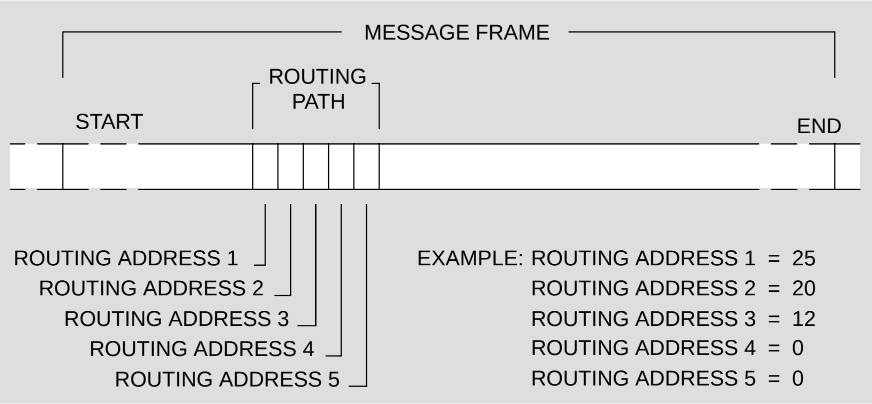


**Figure 2 Message Frame Routing Path**

The example in Figure 2 shows routing to a controller through three networks that are joined by a pair of Bridge Plus devices. Using the routing bytes in the example, the message will be sent first to node 25, a Bridge Plus on the local network. That bridge forwards the message to a second Bridge Plus at node address 20 on the second network. The second bridge forwards the message to its final destination, a controller at node address 12 on the third network. The zero contents of bytes 4 and 5 specify that no further routing will occur.

The routing path contents are specific to the type of device at the destination. Routing path methods for various networked devices are outlined below. For further details about message routing paths, see your *Modbus Planning and Installation Guide.*

### 1.5.2 Routing to Programmable Controllers

For 984 programmable controllers, including the AT-984 and MC-984, the last nonzero byte in the routing specifies the network node address of the controller (range: 1 ... 64). For example, the path 5.0.0.0.0 specifies a controller node at address 5 on the local network (the network to which the host is attached).

### 1.5.3 Routing to Network Adapters

For host-based network adapters such as the SA85 and SM85, the bytes up to and including the adapter's own node address specify the routing to the adapter (e.g., through Bridge Plus devices). The byte immediately following the adapter's node address specifies an internal path or task (range: 1 ... 8) to which the message is to be assigned.

Any remaining bytes after the task byte are available to the user for custom application within the host program (range: 0 ... 255). The adapter does not check the contents of bytes following the task byte.

For example, if an adapter is at node address 35 on the local network, the path 35.8.200.0.0 specifies routing to path 8 in that adapter, with the value 200 used for further application in the host program.

### 1.5.4 Routing to Bridge Multiplexers

For BM85 bridge multiplexers, the routing field contents are specific to the slave device configuration at the multiplexer's Modbus port. Either a single slave device or a network of slave devices can be connected at the port.

A single slave device at a multiplexer's Modbus port is addressed using two bytes. The next-to-last nonzero byte addresses the multiplexer node (range: 1 ... 64). The last nonzero byte specifies the port (range: 1 ... 4) to which the slave device is attached. Specifying the port automatically addresses the device at that port. For example, if a BM85 is at node address 25 on the local network, 25.1.0.0.0 routes a message to the single slave device at the multiplexer's port 1.

A networked slave device at the multiplexer's port is addressed using three bytes. The third-from-last nonzerobyte addresses the multiplexernode (range: 1 ... 64). The next-to-last nonzero byte specifies the port (range: 1 ... 4) to which the network is attached. The last nonzero byte specifies the Modbus address of the slave device (range: 1 ... 247). For example, 25.2.200.0.0 routes a message to multiplexer node address 25, port 2, slave device 200.

## 1.6 Modbus Plus Transactions

With multiple node devices processing messages asynchronously on the network, an individual device might have several concurrent transactions in process. Each device has multiple internal paths of various types to allow concurrent processing of transactions. It opens a path when a transaction begins, keeps it open during processing of the transaction, and closes it when the transaction terminates. When the path is closed, it becomes available to another transaction.

Both the originating and destination devices open paths for a mutual transaction, and maintain the paths until the transaction completes. If the transaction passes through Bridge Plus devices to a destination on another network, each bridge opens and maintains a path at each of its two network ports. Thus a logical path is established between the originating and destination devices, and maintained until the transaction is finished. When the transaction is completed, all of the paths it has used will be freed.

### 1.6.1 Path Types

Each Modbus Plus device contains the following types of paths:

**Data Master (DM) Path** This type of path is opened for data reads and writes, and for get and clear remote statistics, as they are originated in the device. DM paths are identified by a path value in the range 01...08 hexadecimal in programming functions that require the specification of a path.

**Data Slave (DS) Path** This type of path is opened for data reads and writes as they are received by the device. DS paths are identified in the range 41...48 hexadecimal.

**Program Master (PM) Path** This type of path is opened for programming commands as they are originated in the device. PM paths are identified in the range 81...88 hexadecimal.

**Program Slave (PS) Path** This type of path is opened for programming commands as they are received by the device. PS paths are identified in the range C1...C8 hexadecimal.

Each path is independent of the others. Activity in one path does not affect the performance of the other paths.

### 1.6.2 Path Quantities

The following paths are available in the Modbus Plus host based devices:

| | Controllers | BM85 | BP85 | SA85/SM85 |
|---|---|---|---|---|
| Data Master | 8* | 4 | 8 | 8 |
| Data Slave | 4 | 4 | 8 | 8 |
| Program Master | 8 | 4 | 8 | 8 |
| Program Slave | 1 | 4 | 8 | 8 |

* Because the host based controllers have a *virtual network adapter* capability built in, their path quantities are different from other types of 984 controllers - see Section 1.7.

### 1.6.3 Queueing

If all DS paths are active in a device, new incoming transactions will be queued. Transactions will remain queued until a path is available, and will then be removed from the queue and given the path. A final data response will not be returned to the originating application until a full path is available from origin to destination.

When the destination node removes a transaction from its queue, it will wait for the network token and then will request the command again from the originating node. The originator will retransmit the command while the destination retains the token. This process occurs transparently, eliminating the need for polling between the origin and destination devices in the application.

**BP85 Bridge Plus Queueing** Messages which must pass through multiple bridges will be queued (if necessary) within the first bridge, but will not be queued within any subsequent bridges. An attempt to queue in a second bridge will return an error code, which can be tested by the application program in the originating node. This prevents unpredictable delays from queueing across several networks. The originating application can determine how to proceed with outstanding tasks, rather than having to wait through multiple levels of queueing. Tasks that are currently in progress can be allowed to continue, or can be aborted in favor of a higher priority task.

## 1.7 Paths in Host Based Controllers

The AT-984 and MC-984 controllers can communicate with other nodes over the Modbus Plus network, and with the host application. These controllers also have a built-in *virtual network adapter* that allows the host application to communicate directly to the network without passing through the controller's holding registers.

### 1.7.1 Paths Between the Controller and Network

Four controller DM paths are available for MSTR blocks in a ladder logic program. The controller can initiate communication with other nodes using the MSTR. As a path becomes available, it will be given to a waiting MSTR. The controller can also use MSTRs to read/write global data and get/clear network statistics in its peer processor, without the need for a path.

Other nodes on the Modbus Plus network can communicate with the controller over as many as four concurrent DS paths. The nodes can use the single PS path to remotely program the controller.

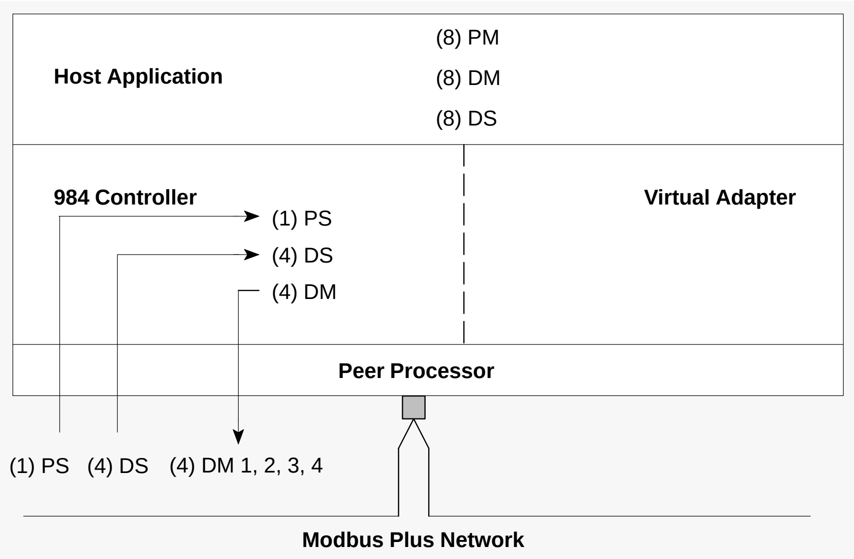


**Figure 3 Paths Between the Controller and Network**

### 1.7.2 Paths Between the Host and Network

The host application can communicate with the peer processor, and therefore with Modbus Plus network nodes, using PM and DM paths through the virtual adapter. Eight PM paths and four DM paths are available for this purpose. This type of communication is timed to occur at the controller's end-of-scan.

Eight host DM paths are present, but paths 1 ... 4 are used only by the controller. DM paths 5 ... 8 may be used directly from the host to the network. Paths in the range 1 ... 4 will be mapped to paths in the range 5 ... 8. If more than four paths are used, the additional paths will be queued until a path 5 ... 8 becomes available.

The host application can read or clear network statistics in the peer processor. It cannot read or write global data directly to the network, but can access holding registers in the controller's logic program that are used for storing the global data.

Remote nodes cannot initiate communications directly with the host application. They can read and write controller registers, which are accessible to the host.

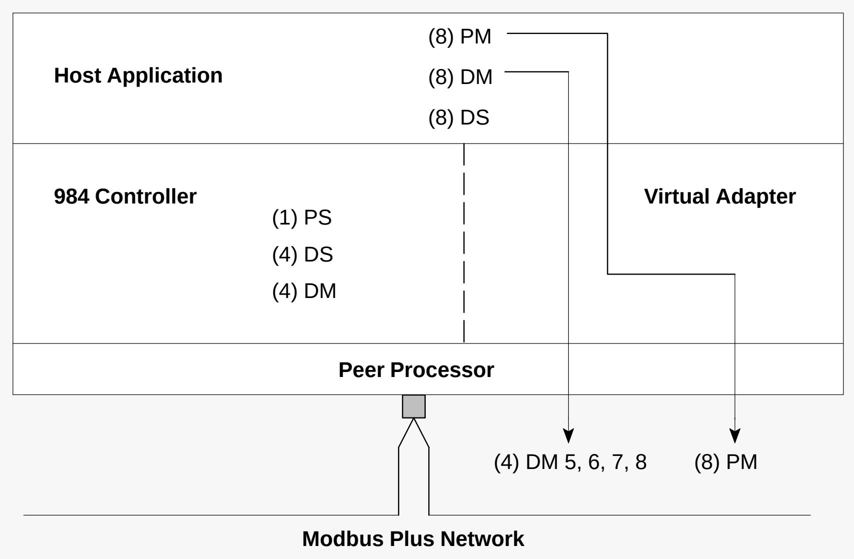


**Figure 4 Paths Between the Host and Network**

### 1.7.3 Paths Between the Host and Controller

The host computer contains eight PM paths, while the controller has one PS path. The host may have any one of its PM paths logged into the 984 at a time.

Eight DM paths are available in the host, while the controller has four DS paths. The host can communicate with the 984 using up to four concurrent DM paths. Any additional host DM tasks will be queued. The host addresses the controller by using its network address in the first byte of the routing path (see Figure 2).

Four controller DM paths are available for MSTR blocks in a ladder logic program. The controller can communicate with its host over these paths, using the host's DS paths. As a path becomes available, it will be given to a waiting MSTR.

The controller addresses its host by using the first two bytes in the routing path (see Figure 2). Routing byte 1 is the Modbus Plus network node address for the local controller. Routing byte 2 is a host DS path number, which can be used to direct the message to an application task (range 1 ... 8) running in the host.

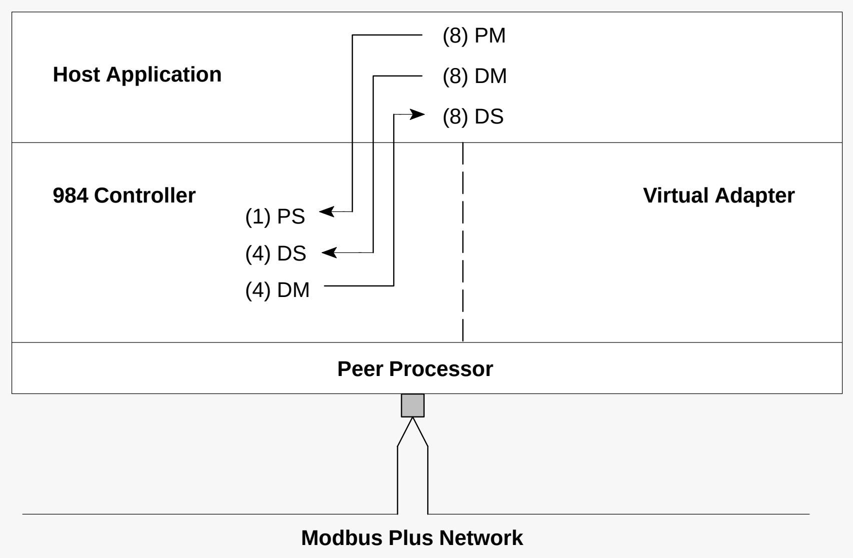


**Figure 5 Paths Between the Host and Controller**

## 1.8 Modbus Data Access Commands

Transactions to or from programmable controller nodes are based on Modbus data access commands that are imbedded into Modbus Plus frames. These commands are recognized by controllers for reading and writing coils and registers, and for reporting status. The following Modbus commands are used:

| Function Code (Decimal) | Command Name |
|---|---|
| 1 | Read Discrete Output Status (0*xxxx*) |
| 2 | Read Discrete Input Status (1*xxxx*) |
| 3 | Read Output Register (4*xxxx*) |
| 4 | Read Input Register (3*xxxx*) |
| 5 | Force Single Coil (0*xxxx*) |
| 6 | Preset Single Register (4*xxxx*) |
| 7 | Read Exception Status |
| 8* | Get/Clear Network Statistics (Subfunction 21) |
| 15 | Force Multiple Coils (0*xxxx*) |
| 16 | Preset Multiple Registers (4*xxxx*) |
| 17 | Report Slave ID |

* Use only subfunction 21 of function 8 for Modbus Plus networking data.

### Path Requirements

All of the Modbus data access commands require a Data Master path in the initiating node. Section 1.6.2 lists the path quantities that are available in Modbus Plus devices.

### Sample Programs

The sample programs on your distribution disks provide examples of how Modbus commands can be imbedded into messages to programmable controllers. The programs also show how to handle the responses from the controllers. Source code and executable files are provided.

For example, the sample program READNODE.EXE reads a controller's discrete inputs, coils, input registers, and holding registers, and displays their contents. You can examine the source file READNODE.C for programming examples.

## 1.9 More Programming Information

Modbus Plus applications can be programmed using Modicon's implementation of NetBIOS commands, described in Appendixes A through C.

Instructions for running the Modbus Plus Network Diagnostic Utility program and for interpreting the network activity are contained in Appendix D.

Instructions for running the sample programs can be found in Appendix E. Your disks contain the full source code files for these programs.

For more details about using Modbus data access commands, see Appendix F.

If you would like further information about the application of your Modbus Plus network, refer to the *Modbus Plus Planning and Installation Guide.* For more information about the 984 Programmable Controller system, see the *984 Controller Systems Manual.*

# Chapter 2
# The AT-984 Controller

# 2.1 AT–984 Overview

The AT-984 board is a full-function 984 programmable controller designed to operate in an IBM-AT or compatible host computer. It is a full-length AT card that resides in one option slot in the host's motherboard. The host communicates directly with the controller over the AT bus. The AT-984 also acts as a Modbus Plus network node for applications running on the host PC—it can send messages generated by its host out over the Modbus Plus network.

## 2.1.1 984 Controller Capabilities

The AT-984 Controller contains a 16-bit word CPU, solving user logic at a rate of 1.5 ms/K words. Models AM-0984-AT0 and -AT2 provide 16K words of user memory, and support up to 7 remote drops of I/O modules. Model AM-0984-AT4 provides 32K words of user memory, and supports up to 16 remote I/O drops. All models contain an S908 Remote I/O Processor that communicates with the remote I/O system at 1.544 Mbaud.

The AT-984 supports drops of Modicon 800, 200, and 500 Series I/O modules. Each drop provides up to 512 bitsin/ 512 bitsout, with up to 2048 bits systemwide. Each drop can support one of the following types of remote I/O interface devices:

| RIO Drop Interface | I/O Module Series | Remote ASCII Support |
|---|---|---|
| J890 | 800 | None |
| J892 | 800 | Two/drop |
| P890 | 800 | None |
| P892 | 800 | Two/drop |
| P451 with J291 | 200 | None |
| P453 with J290 | 200 | Two/drop |
| P451 with J291 and J540 | 500 | None |
| P453 with J290 and J540 | 500 | Two/drop |

An F-connector at the rear panel of the board provides the remote I/O connection. The RIO cable system can operate with up to 32 dB total signal loss, including the losses for all cables, taps, and connectors.

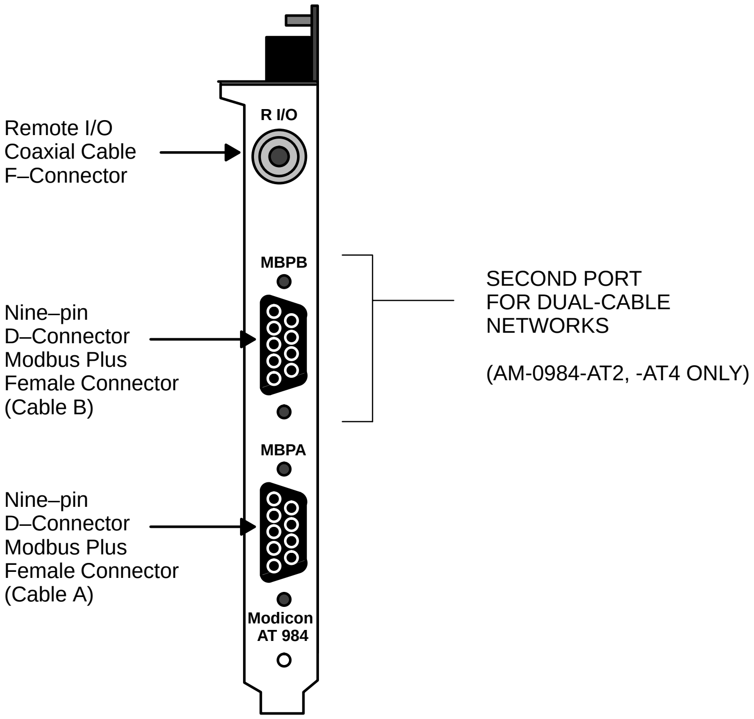


**Figure 6 AT–984 Rear Panel View**

## 2.1.2 Communications Capabilities

The AT-984 Controller communicates with its host PC over the AT bus and can communicate with other devices on a Modbus Plus network via the female nine-pin D-connector on the rear panel of the board. For dual-cable networks, the upper D-connector is for cable B (on AM-0984-A T2, AT4 only).

An AT-984 Controller is supplied with device drivers for DOS and OS/2. The drivers provide an interface to Modicon's implementation of NetBIOS commands.

## 2.1.3 System Planning

For further information about planning your Modbus Plus network system, see the *Modibus Plus Network Planning and Installation Guide.*

## 2.2 Installation Overview

Installation of the AT-984 Controller board consists of five types of actions:

□ Setting the controller's Modbus Plus address switches (see Section 2.4)

□ Setting the controller's memory base address switches (see Section 2.5)

□ Installing the controller's battery and setting its jumpers (see Section 2.6)

□ Installing the controller board into the host and connecting it to the Modbus Plus network (see Section 2.7)

□ Installing the device driver, software library, network diagnostic, and sample programs. Instructions for installing the DOS and OS/2 versions of your software are provided in separate chapters of this guide.

☞ **Note:** Before installing the AT-984, you should be familiar with methods for handling circuit boards, including methods for antistatic protection. If you are not familiar with these methods, contact Modicon for assistance.

### 2.2.1 Adding or Deleting Active Nodes

If you are replacing an AT-984 as a node device on an active Modbus Plus network, you can disconnect the device's local network cable and reconnect it without powering down the devices at the network's other nodes. The network protocol will bypass the removed device, and will include it when reconnected.

⚠ **Caution: If your application is dependent upon the presence of this controller node on the network, adding or deleting it as an active node can produce unpredictable results. Make sure to determine how the application will handle the network configuration before adding or deleting any node.**

Before replacing an AT-984 in your application, make sure its network address switches, memory address switches, and jumpers are set correctly.

If you disconnect a node device from the network, it is not necessary to terminate its local drop connector. The connector should be left open electrically. Cover its pins to prevent damage and contamination.

## 2.3 Locating the Switches

The AT-984 board layout is outlined in Figure 7. Note the locations of the Modbus Plus node address switches and memory window base address switches.

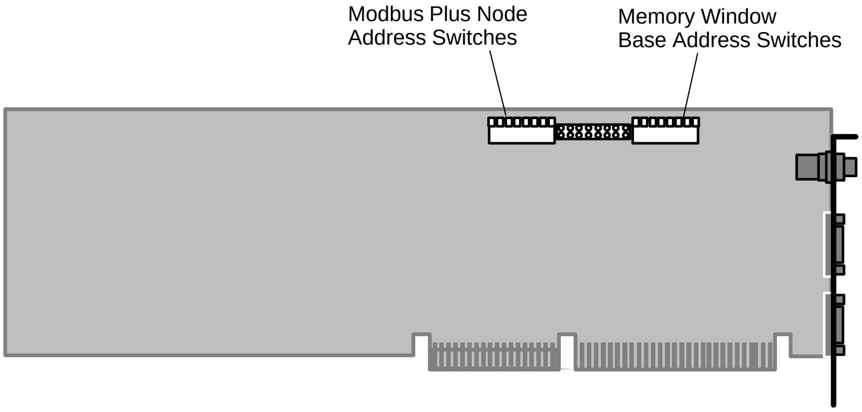
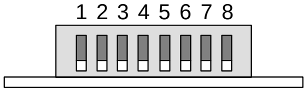


**Figure 7 AT–984 Switch Locations**

## 2.4 Setting the Modbus Plus Address

Set Modbus Plus node address switches 1-6 to the address in your application. Switches 7 and 8 are not used. Each node must have a unique address. Note that the address will be one higher than the binary value you set into the switches.

It is recommended that you reserve address 64 for future network maintenance. It is also recommended that you do not use address 1, to avoid possible confusion when using a local default address of 1 at a controller node's programming panel.



SWITCHES SHOWN IN '0' POSITION
(TOWARD CIRCUIT BOARD)

| | SWITCH POSITION | | | | | | | SWITCH POSITION | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDRESS | 1 | 2 | 3 | 4 | 5 | 6 | ADDRESS | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 33 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 34 | 1 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 | 35 | 0 | 1 | 0 | 0 | 0 | 1 |
| 4 | 1 | 1 | 0 | 0 | 0 | 0 | 36 | 1 | 1 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 1 | 0 | 0 | 0 | 37 | 0 | 0 | 1 | 0 | 0 | 1 |
| 6 | 1 | 0 | 1 | 0 | 0 | 0 | 38 | 1 | 0 | 1 | 0 | 0 | 1 |
| 7 | 0 | 1 | 1 | 0 | 0 | 0 | 39 | 0 | 1 | 1 | 0 | 0 | 1 |
| 8 | 1 | 1 | 1 | 0 | 0 | 0 | 40 | 1 | 1 | 1 | 0 | 0 | 1 |
| 9 | 0 | 0 | 0 | 1 | 0 | 0 | 41 | 0 | 0 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 0 | 1 | 0 | 0 | 42 | 1 | 0 | 0 | 1 | 0 | 1 |
| 11 | 0 | 1 | 0 | 1 | 0 | 0 | 43 | 0 | 1 | 0 | 1 | 0 | 1 |
| 12 | 1 | 1 | 0 | 1 | 0 | 0 | 44 | 1 | 1 | 0 | 1 | 0 | 1 |
| 13 | 0 | 0 | 1 | 1 | 0 | 0 | 45 | 0 | 0 | 1 | 1 | 0 | 1 |
| 14 | 1 | 0 | 1 | 1 | 0 | 0 | 46 | 1 | 0 | 1 | 1 | 0 | 1 |
| 15 | 0 | 1 | 1 | 1 | 0 | 0 | 47 | 0 | 1 | 1 | 1 | 0 | 1 |
| 16 | 1 | 1 | 1 | 1 | 0 | 0 | 48 | 1 | 1 | 1 | 1 | 0 | 1 |
| 17 | 0 | 0 | 0 | 0 | 1 | 0 | 49 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 1 | 0 | 0 | 0 | 1 | 0 | 50 | 1 | 0 | 0 | 0 | 1 | 1 |
| 19 | 0 | 1 | 0 | 0 | 1 | 0 | 51 | 0 | 1 | 0 | 0 | 1 | 1 |
| 20 | 1 | 1 | 0 | 0 | 1 | 0 | 52 | 1 | 1 | 0 | 0 | 1 | 1 |
| 21 | 0 | 0 | 1 | 0 | 1 | 0 | 53 | 0 | 0 | 1 | 0 | 1 | 1 |
| 22 | 1 | 0 | 1 | 0 | 1 | 0 | 54 | 1 | 0 | 1 | 0 | 1 | 1 |
| 23 | 0 | 1 | 1 | 0 | 1 | 0 | 55 | 0 | 1 | 1 | 0 | 1 | 1 |
| 24 | 1 | 1 | 1 | 0 | 1 | 0 | 56 | 1 | 1 | 1 | 0 | 1 | 1 |
| 25 | 0 | 0 | 0 | 1 | 1 | 0 | 57 | 0 | 0 | 0 | 1 | 1 | 1 |
| 26 | 1 | 0 | 0 | 1 | 1 | 0 | 58 | 1 | 0 | 0 | 1 | 1 | 1 |
| 27 | 0 | 1 | 0 | 1 | 1 | 0 | 59 | 0 | 1 | 0 | 1 | 1 | 1 |
| 28 | 1 | 1 | 0 | 1 | 1 | 0 | 60 | 1 | 1 | 0 | 1 | 1 | 1 |
| 29 | 0 | 0 | 1 | 1 | 1 | 0 | 61 | 0 | 0 | 1 | 1 | 1 | 1 |
| 30 | 1 | 0 | 1 | 1 | 1 | 0 | 62 | 1 | 0 | 1 | 1 | 1 | 1 |
| 31 | 0 | 1 | 1 | 1 | 1 | 0 | 63 | 0 | 1 | 1 | 1 | 1 | 1 |
| 32 | 1 | 1 | 1 | 1 | 1 | 0 | 64 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 8 AT–984 Modbus Plus Network Address Switch Settings**

## 2.5 Setting the Memory Base Address

Each board uses a memory area in the computer as a buffer for the board's status and message transactions. This base address prevents conflict with other option boards in the computer.

Valid base address settings range from C0000 ... EF800 hexadecimal. The area used in memory is a 2K bytes (800 hex) portion starting at the base address. Refer to your computer's manual to determine available areas of free memory. Select an area that will not be overwritten by your application or by other options. Record the address. You will need it later when you setup your CONFIG.SYS file.

The top part of Figure 9 shows the address bus range from all 0 to all 1, with the portion seen by the board's switches. The bottom part of the figure shows the lowest and highest base addresses in binary and hexadecimal.

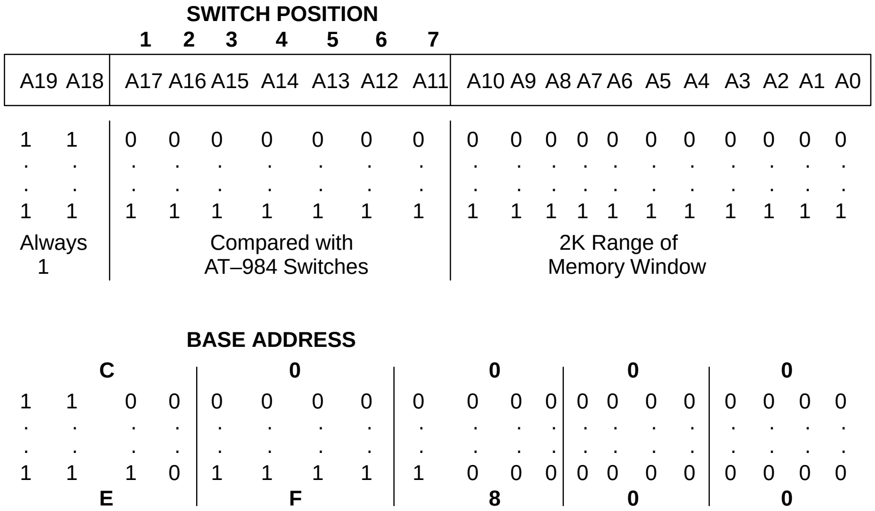
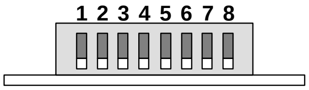


**Figure 9 AT–984 Memory Window Addressing Method**

To decode a memory address, the AT-984 compares the computer's address bus bits A19 and A18 with logic 1's. Bits A17 ... A11 are compared with switch settings on the board. The board is selected when an address matches bits A19 ... A11. Bits A19 ... A11 thus define the base address to be accessed by the application software. Locations within the 2K range are addressed by bits A10 ... A0.

Set the memory base address switches (location on the board shown in Figure 7) to define the base address. Switch 8 is not used.



SWITCHES SHOWN IN ‘0’ POSITION
(TOWARD CIRCUIT BOARD)

| BASE | SWITCH POSITION | | | | | | | BASE | SWITCH POSITION | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDRESS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ADDRESS | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| C0000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | D2800 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| C0800 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D3000 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| C1000 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | D3800 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| C1800 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | D4000 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| C2000 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | D4800 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| C2800 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | D5000 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| C3000 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | D5800 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| C3800 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | D6000 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| C4000 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | D6800 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| C4800 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | D7000 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| C5000 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | D7800 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| C5800 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | D8000 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| C6000 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | D8800 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| C6800 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | D9000 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| C7000 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | D9800 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| C7800 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | DA000 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| C8000 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | DA800 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| C8800 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | DB000 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| C9000 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | DB800 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| C9800 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | DC000 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| CA000 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | DC800 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| CA800 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | DD000 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| CB000 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | DD800 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| CB800 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | DE000 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| CC000 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | DE800 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| CC800 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | DF000 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| CD000 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | DF800 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| CD800 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | E0000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| CE000 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | E0800 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| CE800 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | E1000 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| CF000 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | E1800 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| CF800 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | . . . | . | . | . | . | . | . | . |
| D0000 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | . . . | . | . | . | . | . | . | . |
| D0800 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | EE000 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| D1000 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | EE800 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| D1800 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | EF000 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| D2000 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | EF800 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

**Figure 10 AT–984 Memory Base Address Switch Settings**

## 2.6 Installing the Battery and Jumpers

Before installing the AT-984, you must install its battery and set or verify its jumper positions. Figure 11 shows the locations of the battery and jumpers.

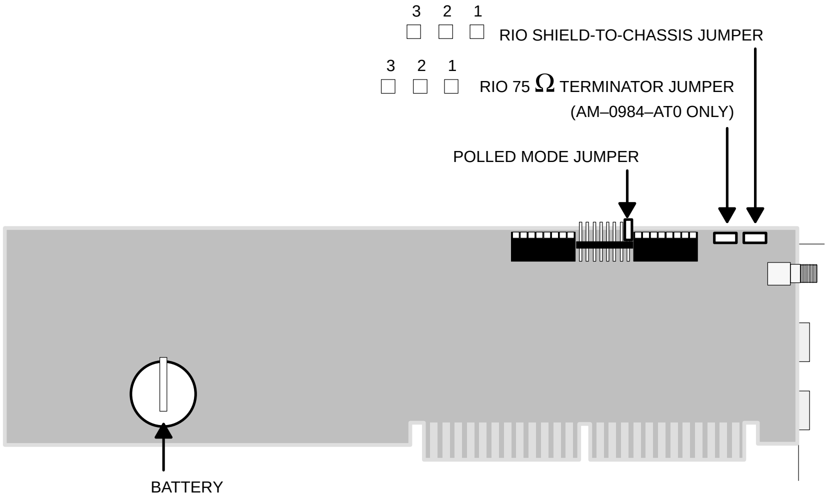


**Figure 11 AT–984 Battery and Jumpers**

### 2.6.1 Installing the Battery

A 2430 type coin cell lithium battery is shipped uninstalled with the AT-984 board. The battery should be inserted into the AT-984 before the board is installed in the host motherboard.

To install the battery, slide it under the lever in the battery holder. When installing the battery, the **+** pole must be facing outward.

### 2.6.2 Polled Mode Jumper

Eight columns of jumper pins, two pins/column, are located between the two DIP switches on the board.

The proper jumper position is across the rightmost column to specify Polled Mode. Figure 12 shows the correct jumper position. Verify that it is in this position. It should be left in this position at all times.

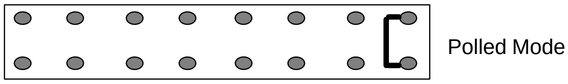


**Figure 12 AT–984 Polled Mode Jumper Setting**

### 2.6.3 RIO Terminator Jumper

AM-0984-A T0 has a jumper for enabling or disabling the 75 Ω terminator for the RIO cable connection. On the AM-0984-AT2 and AM-0984-A T4, the terminator is always connected. If present, the jumper uses a three-pin connector. The board is shipped with the jumper in the 2-3 position.

□ When the jumper is in the 2-3 position, the terminator resistor is *connected*.

□ When the jumper is in the 1-2 position, the terminator resistor is *disconnected*.

Set the jumper to the position that will be used for your application.

### 2.6.4 RIO Shield–to–Chassis Jumper

The RIO cable shield-to-chassis jumper uses a three-pin connector. The board is shipped with the jumper in the 2-3 position.

□ When the jumper is in the 2-3 position, the RIO cable shield is *isolated* from chassis ground by a capacitor.

□ When the jumper is in the 1-2 position, the RIO cable shield is *connected* directly to chassis ground.

Set the jumper to the position that will be used for your application.

### 2.6.5 Other Jumpers

All other jumpers on the board should be left as set by the factory.

## 2.7 Installing the AT–984 Board

After you have set the board's Modbus Plus node address switches and memory base address switches, installed its battery, and set or verified its jumpers, you can install the board in your host computer. Use the guidelines below.

☞ **Note:** Before installing the AT-984, you should be familiar with methods for handling circuit boards, including methods for antistatic protection. If you are not familiar with these methods, contact Modicon for assistance.

**Step 1** Referring to your computer's product documentation, set the host computer power switch to OFF, and unplug its power cable from the power source.

**Step 2** Remove the computer cover, setting aside the bolts and other hardware for later reassembly.

**Step 3** Locate an unused expansion slot connector on the computer motherboard. At the rear of the computer chassis, remove the bolt that secures the blank faceplate at this slot position, and remove the faceplate. Retain the bolt for later reassembly.

**Step 4** Insert the AT-984 board in the expansion slot connector and firmly seat it in the bus connector.

**Step 5** Install the bolt to secure the board's rear faceplate to the computer frame.

☞ **Note:** This bolt is required for proper grounding of the board.

**Step 6** Plug the Modbus Plus network cable connector(s) into the board's connector(s). If you have a dual-cable network, your two cables should be labeled A and B. Make sure to connect the cables into the proper connectors (A and B). Secure each connector by tightening its two screws.

**Step 7** Attach the Remote I/O drop cable to the RIO F-connector on the AT-984 adapter plate.

**Step 8** Reconnect the computer power cable, and power up the computer, and verify normal operation with the board installed. The MODBUS PLUS LED will flash a repetitive pattern indicating the status of the node on the network. Figure 13 shows the location of the LED. Section 2.8 describes the meaning of the LED patterns.

**Step 9** Reinstall the computer cover. The LED displays are not visible with the cover on, so you may want to remove the cover again to view the RUN LED when you implement your user logic application program on the AT-984. Figure 13 shows the location of the LED.

**Step 10** Install the software device driver for your operating system, and install the remaining software files. Instructions are provided in separate DOS and OS/2 chapters of this guide.

## 2.8 Reading the Network Indicator

**AT–0984–AT0**
This board has two green indicators showing the controller's RUN status and Modbus Plus communication status. The indicators are accessible with the computer's top cover removed.

**AT–0984–AT2, –AT4**
This board has two green indicators showing the controller's RUN status and Modbus Plus communication status, and two red indicators which identify faults on the two cable paths. The indicators are accessible with the computer's top cover removed.

If the A or B cable indicator blinks momentarily, it indicates that a message error was detected on the cable path. A steady ON state indicates a hard fault either in the cable or in a node device connected to the cable. If communication is lost on one path, the other path continues normally.

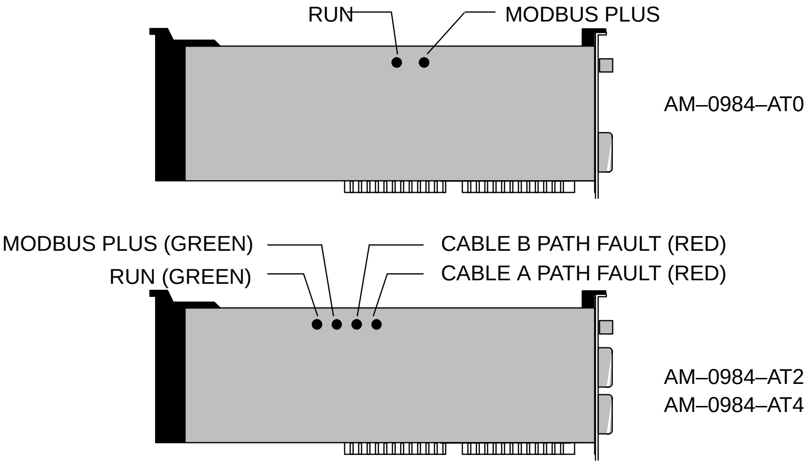


**Figure 13 AT–984 LED Indicator**s

Modbus Plus status is shown by flashing a repetitive pattern on the network indicator. The patterns are:

| LED Pattern | Indication (Status) |
|---|---|
| Six flashes/second | Normal operating state for the node—it is successfully receiving and passing the token. All nodes on a healthy network flash this pattern. |
| One flash/second | The node is off-line just after power-up or after exiting the four flashes/second mode. In this state, the node monitors the network and builds a table of active nodes and token-holding nodes. After being in this state for 5 seconds, the node attempts to go to its normal operating state (indicated by 6 flashes/second). |
| Two flashes, then OFF for two seconds | The node hears the token being passed among the other nodes, but it never receives the token itself—check the network for an open circuit or defective termination. |
| Three flashes, then OFF for 1.7 seconds | The node is not hearing token passing among the other nodes. It periodically claims the token but cannot find another node to which to pass it. Check the network for an open circuit or defective termination. |
| Four flashes, then OFF for 1.4 seconds | The node has heard a valid message from a node using a network address identical to its own address. The node remains in this state for as long as it continues to hear the duplicate address. If the duplicate address is not heard for 5 seconds, the node changes to one flash/second mode. |

### 2.8.1 Network Diagnosis With MBPSTAT

Rather than viewing your network indicator, you may find it more convenient to diagnose suspected faults using your Network Diagnostic Utility program, MBPSTAT.EXE. This utility is supplied on the distribution disk with your controller.

A full description of how to run your MBPSTAT program, and how to select its options for diagnosing your network, is in Appendix D. If you select option 10, ‘Show Node Personality’, your screen will display the same kind of status information that is shown by the flashing patterns of your network indicator. Status is shown in the ‘Peer Status’ line of your MBPSTAT screen.

Here is how your MBPSTAT screen messages correspond to the indicator patterns:

| MBPSTAT Message | Indicator Pattern |
|---|---|
| Normal Link Operation | Six flashes per second |
| Monitor Link Operation | One flash per second |
| Never Getting Token | Two Flashes, then OFF for two seconds |
| Sole Station | Three flashes, then OFF for 1.7 seconds |
| Duplicate Station | Four flashes, then OFF for 1.4 seconds |

## 2.9 Stopped Error Codes

All 984 controllers contain the following stopped error codes, which will be displayed in the panel software:

| Error Code | Mnemonic | Meaning |
|---|---|---|
| 0*x*7FFF | PCSICK | Controller unhealthy |
| 0*x*8000 | PCSTOPPED | Controller stopped |
| 0*x*4000 | BADTCOP | Bad I/O traffic cop table |
| 0*x*2000 | DIMAWAR | PLC in DIM AWARENESS state |
| 0*x*1000 | PORTIVENT | Bad port intervention |
| 0*x*0800 | BADSEGSCH | Bad segment scheduler |
| 0*x*0400 | SONNOTIST | Start of network (SON) did not start segment |
| 0*x*0200 | PDCHEKSUM | Bad power–down checksum |
| 0*x*0080 | NOEOLDOIO | Watchdog timer has expired |
| 0*x*0040 | RTCFAILED | Real time clock failure |
| 0*x*0020 | BADOXUSED | Bad *coil used* table |
| 0*x*0010 | RIOFAILED | Remote I/O failure |
| 0*x*0008 | NODETYPE | Illegal node type used |
| 0*x*0004 | ULCSUMERR | User logic checksum error |
| 0*x*0002 | DSCRDISAB | Discrete disable error |
| 0*x*0001 | BADCONFIG | Bad configuration table |

Stopped error states of various controller nodes may also be sent over the Modbus Plus network using Modbus function 11 hex, as described in Appendix F.

## 2.10 Labeling the Modbus Plus Port

Two sets of labels are provided with your AT-984 Controller card to identify the network and node address at its Modbus Plus port. One label should be attached to the unit when you complete the connection to the network. The other is a spare.

Enter onto the label the Modbus Plus network number and node address you have assigned to the unit's Modbus Plus port. Place the label on the unit so that it can readily identify the port.

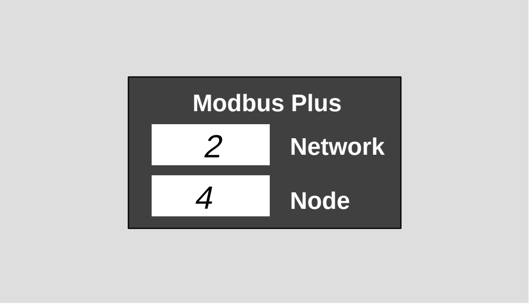


**Figure 14 AT–984 Modbus Plus Port Label**

## 2.11 Initializing the AT–984

During installation of your AT-984, you setup its parameters in a set of swtiches. These include the network address and memory window base address.

If you recycle power to your host computer, the AT-984 will apply the parameters as they are currently set. If you do a keyboard ‘warm boot’ with Ctrl-Alt-Del, the parameters will not be affected. If you want to change any parameters, you must recycle power.

During a ‘warm boot’, the device will continue with Modbus Plus activity, including passing tokens and handling any transactions that are active. If the device is solving ladder logic, it will continue doing so without interruption.

# 2.12 AT–984 Specifications

**AT–984 Controller Specifications**

| | | |
|---|---|---|
| Description | Name | AT–984 Programmable Controller with Modubs Plus |
| | Part Number | AM–0984–AT0 (Single Cable)<br>AM–0984–AT2, –AT4 (Dual Cables) |
| Physical Characteristics | Size | Standard Full Slot Board,<br>13.3 x 4.5 in (337.6 x 114.6 mm) |
| | Weight | 1.0 lb (0.45 kg) net |
| | | 2.0 lbs (0.9 kg) shipping |
| Power | Operating Current | From Computer Motherboard,<br>750 mA typical; 1.1 A maximum |
| Environmental | Temperature | 0 ... 60 degrees C, operating |
| | | –40 ... +80 degrees C, storage |
| | Humidity | 0 ... 95%, non–condensing |
| | EMI, Radiated Susceptibility | MIL STD 461B RS02, RS03 |
| | EMI, Conducted Susceptibility | MIL STD 461B CS02 |
| Network Connection | Connector Type | Mates with Modbus Plus drop cable |
| Software | Operating System | MS–DOS 3.1 or later<br>OS/2 1.0 or later |
| | C Library | Microsoft C 5.1 (large model) |
| | DESQview | Version 2.24 |
| User Memory | Size | AM–0984–AT0, –AT2: 16K Words<br>AM–094–AT4: 32K Words |
| I/O Capability | RIO Connector | F–Connector for RG6/U RIO Cable |
| | RIO Communication | S908 Protocol |
| | Logic Solve Time | 1.5 ms/K words of user logic |
| | RIO Cable System Loss | Up to 32 dB loss |
| | RIO Drops Supported | AM–0984–AT0, –AT2: Up to 7 drops<br>AM–0984–AT4: Up to 16 drops |
| | Local Drops Supported | None |
| | ASCII Devices Supported (2 per drop) | AM–0984–AT0, –AT2: Up to 14<br>AM–0984–AT4: Up to 32 |

# Chapter 3
# The SA85 Network Adapter

## 3.1 The SA85 and Your Computer

### 3.1.1 Your Hardware Configuration

Before installing the SA85 board, you'll assign its network node address and memory window base address in a set of hardware switches. The node address identifies the SA85 for tokens and messages on the Modbus Plus network. The memory address defines a 2K bytes area in your computer that will be used as a buffer between the SA85 and your application.

You'll also need to verify the factory setting of the board's polled mode jumper. This jumper and all other jumpers are preset and are not configurable by the user.

You can then install the unit into an available slot in your computer's motherboard, and connect the network cable.

### 3.1.2 Your Software Configuration

Before using the SA85 in your application, you must install its device driver on your hard disk and edit a command line into your CONFIG.SYS file. This will assign an adapter number, memory window base address, and software interrupt to the driver. These parameters will identify the SA85 uniquely to your application, even when multiple options may be present in your computer. Separate drivers are supplied for DOS and OS/2.

You can install the source code, headers, and library files supplied with your SA85. You can compile and link them to your application program using the Microsoft C compiler.

You also have a network diagnostic utility and a set of sample programs that show methods for accessing controller registers and the network's global database.

Figure 15 summarizes the configuration of the SA85 in your computer product.

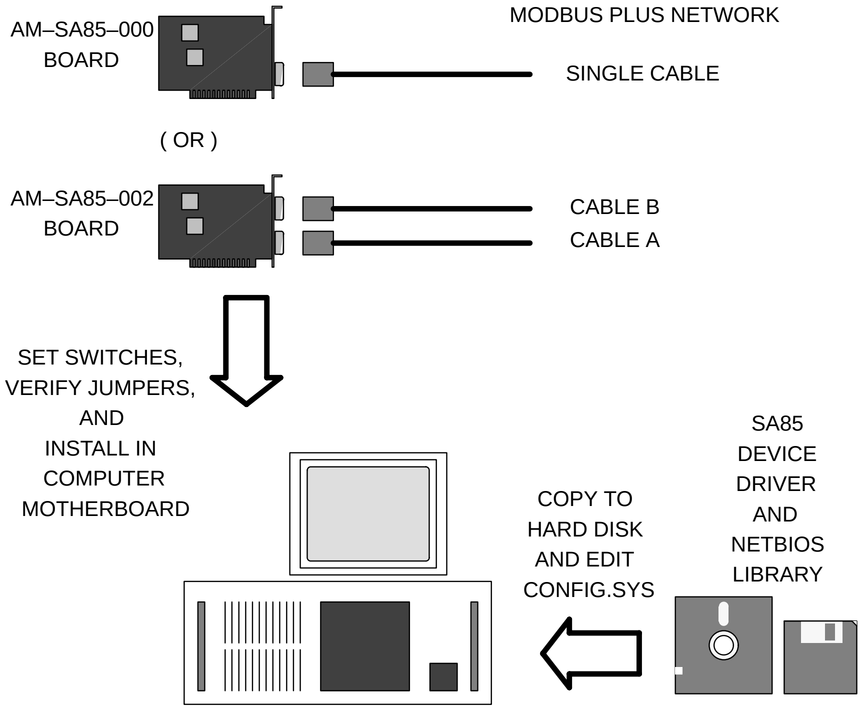


**Figure 15 Overview of the SA85 and Host Configuration**

### 3.1.3 System Planning

For further information about planning your Modbus Plus network system, see the *Modibus Plus Network Planning and Installation Guide.*

## 3.2 Installation Overview

Installation of the SA85 Network Adapter consists of five kinds of actions:

□ Setting the adapter's Modbus Plus address switches (see Section 3.4)

□ Setting the adapter's memory base address switches (see Section 3.5)

□ Verifying the adapter's jumpers (see Section 3.6)

□ Installing the adapter board into the host and connecting it to the Modbus Plus network (see Section 3.7)

□ Installing the device driver, software library, network diagnostic, and sample programs. Instructions for installing the DOS and OS/2 versions of your software are provided in separate chapters of this guide.

☞ **Note:** Before installing the SA85, you should be familiar with methods for handling circuit boards, including methods for antistatic protection. If you are not familiar with these methods, contact Modicon for assistance.

### 3.2.1 Adding or Deleting Active Nodes

If you are replacing an SA85 as a node device on an active Modbus Plus network, you can disconnect the device's local drop cable and reconnect it without powering down the devices at the network's other nodes. The network protocol will bypass the removed device, and will include it when reconnected.

**Caution: If your application is dependent upon the presence of this adapter node on the network, adding or deleting it as an active node can produce unpredictable results. Make sure to determine how the application will handle the network configuration before adding or deleting any node.**

Before replacing an SA85 in your application, make sure its network address switches, memory address switches, and jumpers are set correctly.

If you disconnect a node device from the network, it is not necessary to terminate its local drop connector. The connector should be left open electrically. Cover its pins to prevent damage and contamination.

## 3.3 Locating the Switches

The two SA85 board layouts are shown below. Note the locations of the Modbus Plus network address switches and memory window base address switches.

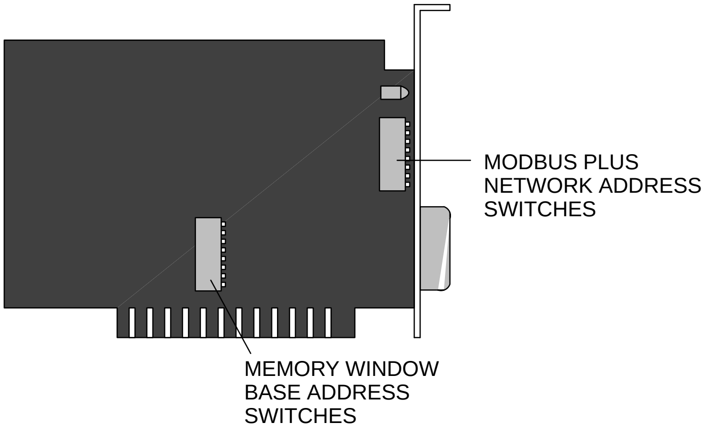

MODBUS PLUS NETWORK ADDRESS SWITCHES
MEMORY WINDOW BASE ADDRESS SWITCHES

**Figure 16 AM–SA85–000 Switch Locations**

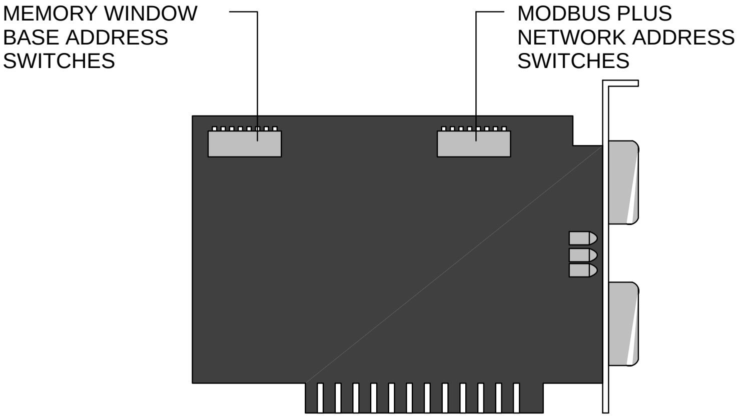
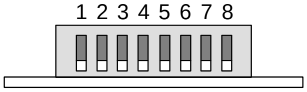

MEMORY WINDOW BASE ADDRESS SWITCHES
MODBUS PLUS NETWORK ADDRESS SWITCHES

**Figure 17 AM–SA85–002 Switch Locations**

# 3.4 Setting the Modbus Plus Address

Set Modbus Plus node address switches 1-6 to the address in your application. Switches 7 and 8 are not used. Each node must have a unique address. Note that the address will be one higher than the binary value you set into the switches.

It is recommended that you reserve address 64 for future network maintenance. It is also recommended that you do not use address 1, to avoid possible confusion when using a local default address of 1 at a controller node's programming panel.



SWITCHES SHOWN IN '0' POSITION
(TOWARD CIRCUIT BOARD)

| | SWITCH POSITION | | | | | | | SWITCH POSITION | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDRESS | 1 | 2 | 3 | 4 | 5 | 6 | ADDRESS | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 33 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 34 | 1 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 | 35 | 0 | 1 | 0 | 0 | 0 | 1 |
| 4 | 1 | 1 | 0 | 0 | 0 | 0 | 36 | 1 | 1 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 1 | 0 | 0 | 0 | 37 | 0 | 0 | 1 | 0 | 0 | 1 |
| 6 | 1 | 0 | 1 | 0 | 0 | 0 | 38 | 1 | 0 | 1 | 0 | 0 | 1 |
| 7 | 0 | 1 | 1 | 0 | 0 | 0 | 39 | 0 | 1 | 1 | 0 | 0 | 1 |
| 8 | 1 | 1 | 1 | 0 | 0 | 0 | 40 | 1 | 1 | 1 | 0 | 0 | 1 |
| 9 | 0 | 0 | 0 | 1 | 0 | 0 | 41 | 0 | 0 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 0 | 1 | 0 | 0 | 42 | 1 | 0 | 0 | 1 | 0 | 1 |
| 11 | 0 | 1 | 0 | 1 | 0 | 0 | 43 | 0 | 1 | 0 | 1 | 0 | 1 |
| 12 | 1 | 1 | 0 | 1 | 0 | 0 | 44 | 1 | 1 | 0 | 1 | 0 | 1 |
| 13 | 0 | 0 | 1 | 1 | 0 | 0 | 45 | 0 | 0 | 1 | 1 | 0 | 1 |
| 14 | 1 | 0 | 1 | 1 | 0 | 0 | 46 | 1 | 0 | 1 | 1 | 0 | 1 |
| 15 | 0 | 1 | 1 | 1 | 0 | 0 | 47 | 0 | 1 | 1 | 1 | 0 | 1 |
| 16 | 1 | 1 | 1 | 1 | 0 | 0 | 48 | 1 | 1 | 1 | 1 | 0 | 1 |
| 17 | 0 | 0 | 0 | 0 | 1 | 0 | 49 | 0 | 0 | 0 | 0 | 1 | 1 |
| 18 | 1 | 0 | 0 | 0 | 1 | 0 | 50 | 1 | 0 | 0 | 0 | 1 | 1 |
| 19 | 0 | 1 | 0 | 0 | 1 | 0 | 51 | 0 | 1 | 0 | 0 | 1 | 1 |
| 20 | 1 | 1 | 0 | 0 | 1 | 0 | 52 | 1 | 1 | 0 | 0 | 1 | 1 |
| 21 | 0 | 0 | 1 | 0 | 1 | 0 | 53 | 0 | 0 | 1 | 0 | 1 | 1 |
| 22 | 1 | 0 | 1 | 0 | 1 | 0 | 54 | 1 | 0 | 1 | 0 | 1 | 1 |
| 23 | 0 | 1 | 1 | 0 | 1 | 0 | 55 | 0 | 1 | 1 | 0 | 1 | 1 |
| 24 | 1 | 1 | 1 | 0 | 1 | 0 | 56 | 1 | 1 | 1 | 0 | 1 | 1 |
| 25 | 0 | 0 | 0 | 1 | 1 | 0 | 57 | 0 | 0 | 0 | 1 | 1 | 1 |
| 26 | 1 | 0 | 0 | 1 | 1 | 0 | 58 | 1 | 0 | 0 | 1 | 1 | 1 |
| 27 | 0 | 1 | 0 | 1 | 1 | 0 | 59 | 0 | 1 | 0 | 1 | 1 | 1 |
| 28 | 1 | 1 | 0 | 1 | 1 | 0 | 60 | 1 | 1 | 0 | 1 | 1 | 1 |
| 29 | 0 | 0 | 1 | 1 | 1 | 0 | 61 | 0 | 0 | 1 | 1 | 1 | 1 |
| 30 | 1 | 0 | 1 | 1 | 1 | 0 | 62 | 1 | 0 | 1 | 1 | 1 | 1 |
| 31 | 0 | 1 | 1 | 1 | 1 | 0 | 63 | 0 | 1 | 1 | 1 | 1 | 1 |
| 32 | 1 | 1 | 1 | 1 | 1 | 0 | 64 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 18 SA85 Modbus Plus Network Address Switch Settings**

# 3.5 Setting the Memory Base Address

The SA85 board uses a memory area in your computer as a buffer for the board's status and message transactions. You must define a base address for this memory area that prevents conflict with other option boards in your computer.

Valid base address settings range from C0000 ... EF800 hexadecimal. The area used in memory is a 2K bytes (800 hex) portion starting at the base address. Refer to your computer's manual to determine available areas of free memory. Select an area that will not be overwritten by your application or by other options. Record the address. You will need it later when you setup your CONFIG.SYS file.

The top part of Figure 19 shows the address bus range from all 0 to all 1, with the portion seen by the board's switches. The bottom part of the figure shows the lowest and highest base addresses in binary and hexadecimal.

**SWITCH POSITION**

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A19 | A18 | A17 | A16 | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Always 1 | | Compared with SA85 Switches | | | | | | | 2K Range of Memory Window | | | | | | | | | | |

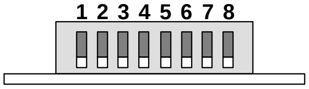
**BASE ADDRESS**

| C | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| E | | | | F | | | | 8 | | | | 0 | | | | 0 | | | |

**Figure 19 SA85 Memory Window Addressing Method**

To decode a memory address, the SA85 compares the computer's address bus bits A19 and A18 with logic 1's. Bits A17 ... A11 are compared with the SA85 switch settings. The board is selected when an address matches bits A19 ... A11. Bits A19 ... A11 thus define the base address to be accessed by the application software. Locations within the 2K range are addressed by bits A10 ... A0.

Set the memory base address switches (location on the board shown in Figure 16 and Figure 17) to define the base address. Switch 8 is not used.



SWITCHES SHOWN IN ‘0’ POSITION
(TOWARD CIRCUIT BOARD)

| BASE | SWITCH POSITION | | | | | | | BASE | SWITCH POSITION | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDRESS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ADDRESS | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| C0000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | D2800 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| C0800 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D3000 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| C1000 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | D3800 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| C1800 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | D4000 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| C2000 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | D4800 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| C2800 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | D5000 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| C3000 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | D5800 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| C3800 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | D6000 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| C4000 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | D6800 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| C4800 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | D7000 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| C5000 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | D7800 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| C5800 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | D8000 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| C6000 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | D8800 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| C6800 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | D9000 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| C7000 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | D9800 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| C7800 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | DA000 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| C8000 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | DA800 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| C8800 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | DB000 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| C9000 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | DB800 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| C9800 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | DC000 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| CA000 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | DC800 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| CA800 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | DD000 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| CB000 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | DD800 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| CB800 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | DE000 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| CC000 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | DE800 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| CC800 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | DF000 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| CD000 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | DF800 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| CD800 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | E0000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| CE000 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | E0800 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| CE800 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | E1000 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| CF000 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | E1800 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| CF800 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | . . . | . | . | . | . | . | . | . |
| D0000 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | . . . | . | . | . | . | . | . | . |
| D0800 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | EE000 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| D1000 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | EE800 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| D1800 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | EF000 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| D2000 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | EF800 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

**Figure 20 SA85 Memory Base Address Switch Settings**

## 3.6 Verifying the Jumpers

### 3.6.1 Polled Mode Jumper

Before installing the SA85 you must verify its Polled Mode jumper setting. Eight rows of jumper pins, two pins/row, are provided on the SA85 board for future use. The pins are located as shown in the two figures below.

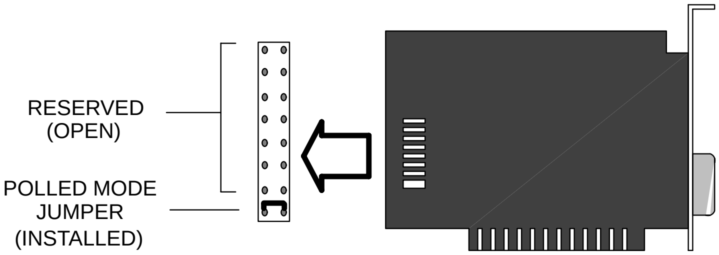


Figure 21 AM–SA85–000 Jumper Setting

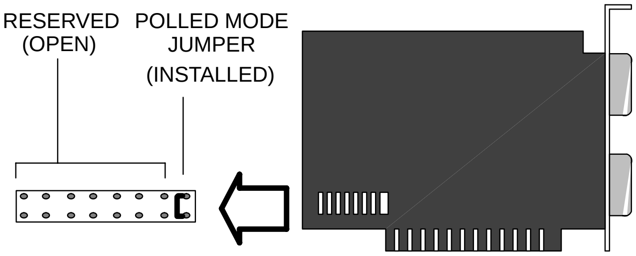


Figure 22 AM–SA85–002 Jumper Setting

The proper jumper position is as shown in the figures, to specify Polled Mode. Verify that it is in this position. It should be left in this position at all times. All other jumpers on the board should be left as set by the factory.

## 3.7 Installing the SA85 Board

Use these guidelines to install the SA85 board and connect it to the network cable:

**Step 1** If you have not set and verified the SA85 network address, memory base address, and jumper, do so now. Refer to the procedures earlier in this chapter to set them.

**Step 2** Referring to your computer's product documentation, set the computer power switch to OFF, and unplug its power cable from the power source.

**Step 3** Remove the computer cover. Retain the bolts and other hardware for later reassembly.

**Step 4** Locate an unused expansion slot connector on the computer motherboard. Remove the bolt securing the blank rear faceplate for this slot position, and remove the faceplate. Retain the bolt for later reassembly.

**Step 5** Insert the SA85 board into the expansion slot connector. Make sure the board is firmly seated in the connector.

**Step 6** Install the bolt to secure the board's rear faceplate to the computer frame.

☞ **Note:** This bolt is required for proper grounding of the board.

**Step 7** Reinstall the computer cover.

**Step 8** Plug the Modbus Plus network cable connector(s) into the board's connector(s). If you have a dual-cable network, your two cables should be labeled A and B. Make sure to connect the cables into the proper connectors (A and B). Secure each connector by tightening its two screws.

**Step 9** Reconnect the computer power cable and power up the computer. Verify normal operation with the board installed.

**Step 10** Install the software device driver for your operating system, and install the remaining software files. Instructions are provided in separate DOS and OS/2 chapters of this guide.

## 3.8 Reading the Network Indicator

**AM–SA85–000**
This board has a single green indicator that shows the network communication status at the SA85 node.

**AM–SA85–002**
This board has three indicators. A green indicator shows the overall communication status at the SA85 node. Two red indicators identify faults on the two cable paths.

If a red indicator blinks momentarily, it indicates that a message error was detected on the cable path. A steady ON state indicates a hard fault either in the cable or in a node device connected to the cable. If communication is lost on one cable path, the other path continues normally.

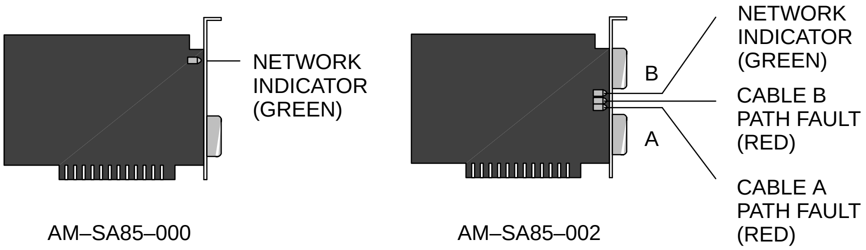


**Figure 23 SA85 Network Indicators**

Modbus Plus status is shown by flashing a repetitive pattern on the network indicator. The patterns are:

| LED Pattern | Indication (Status) |
|---|---|
| Six flashes/second | Normal operating state for the node—it is successfully receiving and passing the token. All nodes on a healthy network flash this pattern. |
| One flash/second | The node is off-line just after power-up or after exiting the four flashes/second mode. In this state, the node monitors the network and builds a table of active nodes and token-holding nodes. After being in this state for 5 seconds, the node attempts to go to its normal operating state (indicated by 6 flashes/second). |
| Two flashes, then OFF for two seconds | The node hears the token being passed among the other nodes, but it never receives the token itself—check the network for an open circuit or defective termination. |
| Three flashes, then OFF for 1.7 seconds | The node is not hearing token passing among the other nodes. It periodically claims the token but cannot find another node to which to pass it. Check the network for an open circuit or defective termination. |
| Four flashes, then OFF for 1.4 seconds | The node has heard a valid message from a node using a network address identical to its own address. The node remains in this state for as long as it continues to hear the duplicate address. If the duplicate address is not heard for 5 seconds, the node changes to one flash/second mode. |

### 3.8.1 Network Diagnosis With MBPSTAT

Rather than viewing your network indicator, you may find it more convenient to diagnose suspected faults using your Network Diagnostic Utility program, MBPSTAT.EXE. This utility is supplied on the distribution disk with your controller.

A full description of how to run your MBPSTAT program, and how to select its options for diagnosing your network, is in Appendix D. If you select option 10, ‘Show Node Personality’, your screen will display the same kind of status information that is shown by the flashing patterns of your network indicator. Status is shown in the ‘Peer Status’ line of your MBPSTAT screen.

Here is how your MBPSTAT screen messages correspond to the indicator patterns:

| MBPSTAT Message | Indicator Pattern |
|---|---|
| Normal Link Operation | Six flashes per second |
| Monitor Link Operation | One flash per second |
| Never Getting Token | Two Flashes, then OFF for two seconds |
| Sole Station | Three flashes, then OFF for 1.7 seconds |
| Duplicate Station | Four flashes, then OFF for 1.4 seconds |

## 3.9 Labeling the Modbus Plus Port

Two sets of labels are provided with the SA85 to identify its Modbus Plus network and node address. One label should be attached to the unit when you complete the connection to the network. The other set is a spare.

Enter onto the label the Modbus Plus network number and node address you have assigned to the SA85. Place the label on the unit so that it can easily be seen. Figure 24 shows an example of the completed label.

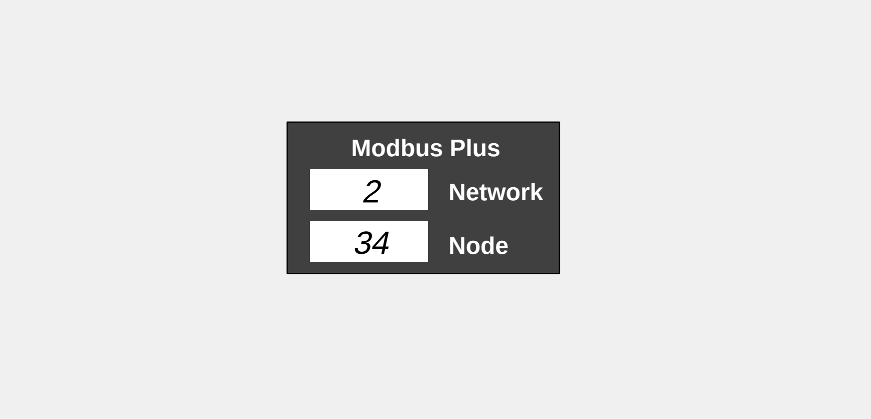


**Figure 24 SA85 Modbus Plus Port Label**

## 3.10 Initializing the SA85

During installation of your SA85, you setup its parameters in a set of swtiches. These include the network address and memory window base address.

If you recycle power to your host computer, the SA85 will apply the parameters as they are currently set. If you do a keyboard ‘warm boot’ with Ctrl-Alt-Del, the parameters will not be affected. If you want to change any parameters, you must recycle power.

During a ‘warm boot’, the device will continue with Modbus Plus activity, including passing tokens and handling any transactions that are active.

# 3.11 SA85 Specifications

**SA85 Network Adapter Specifications**

| | | |
|---|---|---|
| Description | Name | SA85 Modbus Plus Adapter |
| | Part Number | AM–SA85–000 (Single Cable)<br>AM–SA85–002 (Dual Cables) |
| Physical Characteristics | Size | Standard Half Slot Board, 5.2 x 4.2 in (132 x 107 mm) |
| | Weight | 1.0 lb (0.45 kg) net |
| | | 2.0 lbs (0.9 kg) shipping |
| Power | Operating Current | From Computer Motherboard, 500 mA at 5 Vdc maximum |
| Environmental | Temperature | 0 ... 60 degrees C, operating |
| | | –40 ... +80 degrees C, storage |
| | Humidity | 0 ... 95%, non–condensing |
| | EMI, Radiated Susceptibility | MIL STD 461B RS02, RS03 |
| | EMI, Conducted Susceptibility | MIL STD 461B CS02 |
| Network Connection | Connector Type | Mates with Modbus Plus drop cable |
| Software | Operating System | MS–DOS 3.1 or later<br>OS/2 1.0 or later |
| | C Library | Microsoft C 5.1 (large model) |
| | DESQview | Version 2.24 |

# Chapter 4
# The MC-984 Controller

## 4.1 MC-984 Overview

The MC-984 board is a full-function 984 programmable controller designed to operate in an IBM or compatible host computer using Micro Channel architecture. It is a standard Micro Channel card that resides in one slot in the host computer's motherboard. The host communicates directly with the controller over the Micro Channel bus. The MC-984 also acts as a Modbus Plus node for applications running on the host—it can send messages generated by its host out over the Modbus Plus network.

### 4.1.1 984 Controller Capabilities

The MC-984 Controller contains a 16-bit word CPU, provides 16K words of on-board user memory, and can solve user logic at a rate of 1.5 ms/K words. It contains an onboard S908 Remote I/O Processor, supporting up to seven remote drops of I/O modules and communicating with the I/O at 1.544 Mbaud.

The MC-984 supports drops of Modicon 800, 200, and 500 Series I/O modules. Each drop provides up to 512 bitsin/ 512 bitsout, with up to 2048 bits systemwide. Each drop can support two ASCII devices (up to 14 ASCII devices systemwide). Supported remote I/O interface devices include:

| RIO Drop Interface | I/O Module Series | Remote ASCII Support |
|---|---|---|
| J890 | 800 | None |
| J892 | 800 | Two/drop |
| P890 | 800 | None |
| P892 | 800 | Two/drop |
| P451 with J291 | 200 | None |
| P453 with J290 | 200 | Two/drop |
| P451 with J291 and J540 | 500 | None |
| P453 with J290 and J540 | 500 | Two/drop |

An F-connector at the rear panel of the board provides the remote I/O connection. The RIO cable system can operate with up to 32 dB total signal loss, including the losses for all cables, taps, and connectors.

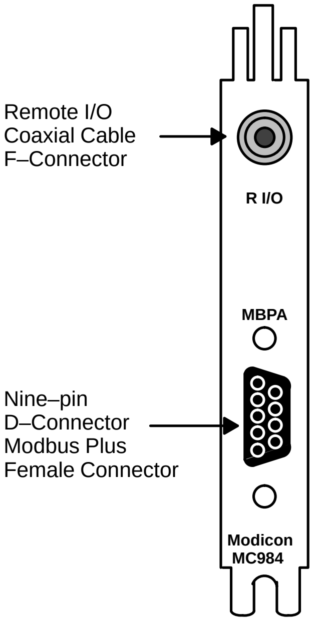


**Figure 25 MC–984 Rear Panel View**

## 4.1.2 Communications Capabilities

The MC-984 Controller communicates with the host computer over the Micro Channel bus and can communicate with other devices on a Modbus Plus network via a female nine-pin D-connector on the rear panel of the board.

An MC-984 Controller is supplied with device drivers for DOS and OS/2. The drivers provide an interface to Modicon's implementation of NetBIOS networking commands.

## 4.1.3 System Planning

For further information about planning your Modbus Plus network system, see the *Modibus Plus Network Planning and Installation Guide.*

## 4.2 Installation Overview

Installation of the MC-984 Controller board consists of five types of actions:

□ Updating the computer's reference disk—letting the system automatically copy the MC-984 option file to the reference disk (see Section 4.3)

□ Installing the controller's battery and setting its jumpers (see Section 4.4)

□ Installing the controller board into the host (see Section 4.5)

□ Configuring the controller's network address, memory window base address, and interrupt, and connecting it to the network (see Section 4.6)

□ Installing the device driver, software library, network diagnostic, and sample programs. Instructions for installing the DOS and OS/2 versions of your software are provided in separate chapters of this guide.

☞ **Note:** Before installing the MC-984, you should be familiar with methods for handling circuit boards, including methods for antistatic protection. If you are not familiar with these methods, contact Modicon for assistance.

### 4.2.1 Adding or Deleting Active Nodes

If you are replacing an MC-984 as a node device on an active Modbus Plus network, you can disconnect the device's local network cable and reconnect it without powering down the devices at the network's other nodes. The network protocol will bypass the removed device, and will include it when reconnected.

⚠ **Caution: If your application is dependent upon the presence of this controller node on the network, adding or deleting it as an active node can produce unpredictable results. Make sure to determine how the application will handle the network configuration before adding or deleting any node.**

If you disconnect a node device from the network, it is not necessary to terminate its local drop connector. The connector should be left open electrically. Cover its pins to prevent damage and contamination.

## 4.3 Updating the Reference Disk

Your computer system includes a reference disk that contains files describing the options that are present. As the first step in your MC-984 installation, you must update the disk to include the MC-984 as a new option. The process is handled by menus when you boot your system.

☞ **Note:** You must update the reference disk before you install the MC-984 into your computer product. If you try to start a computer with the MC-984 installed without updating the reference disk, the system will return `system error 165`.

You will need a working copy of the reference disk supplied with your system—the disk should not be set for write protection. You will also need the diskette 1 from the AS-DIBM-220 software package supplied with your MC-984. Refer to your computer product manuals as required to do the following steps:

**Step 1** Insert the reference disk into your computer's disk drive and boot the computer. The computer's 'logo' identification screen should appear.

**Step 2** Press <ENTER> to clear the logo screen and display the selections menu. Select menu item 5:

```
Copy an option diskette.
```

**Step 3** Following the on-screen instructions, insert your option diskette (AS-DIBM-220, disk 1 of 2), then reinsert the reference disk. The process of updating the reference disk will be handled automatically.

**Step 4** A screen message will appear when the update process is complete. Leave the reference disk in place in the disk drive. Turn OFF your computer's power switch and continue the installation procedure.

# 4.4 Installing the Battery and Jumpers

Before installing the MC-984, you must install its battery and set or verify its jumper positions. Figure 26 shows the locations of the battery and jumpers.

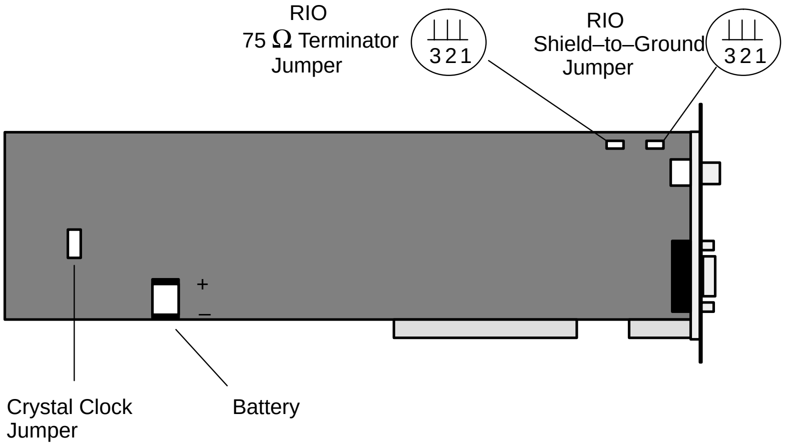


**Figure 26 MC–984 Battery and Jumpers**

## 4.4.1 Installing the Battery

A 1/3N style 3 Volt lithium battery is shipped uninstalled with the MC-984 board. It should be inserted in the MC-984 before the board is installed in the host.

To install the battery, insert it into its holder on the board. When inserting the battery, the **+** pole must be oriented upward and the **-** pole downward as shown in the board view of Figure 26.

## 4.4.2 Crystal Clock Jumper

The crystal clock jumper uses a two-pin connector. Verify that the jumper is in place. It must be left in place at all times.

## 4.4.3 RIO Terminator Jumper

The RIO 75 Ω terminator resistor jumper uses a three-pin connector. The board is shipped with the jumper in the 2-3 position.

□ When the jumper is in the 2-3 position, the terminator resistor is *connected*.

□ When the jumper is in the 1-2 position, the terminator resistor is *disconnected*.

Set the jumper to the position that will be used for your application.

### 4.4.4 RIO Shield–to–Chassis Jumper

The RIO cable shield-to-chassis jumper uses a three-pin connector. The board is shipped with the jumper in the 2-3 position.

□ When the jumper is in the 2-3 position, the RIO cable shield is *isolated* from chassis ground by a capacitor.

□ When the jumper is in the 1-2 position, the RIO cable shield is *connected* directly to chassis ground.

Set the jumper to the position that will be used for your application.

### 4.4.5 Other Settings

The MC-984 Controller board does not contain any user-configurable switches. The board's Modbus Plus node address, memory base address, and interrupt level will be set in the software configuration process after the board is installed into the host's motherboard.

All other jumpers on the board should be left as set by the factory.

## 4.5 Installing the MC–984 Board

Use these guidelines to install the board into your computer, referring to your computer product manuals as required to do the following steps.

**Step 1** Set the computer power switch to OFF, and unplug its power cable from the power source.

**Step 2** Remove the computer cover. Retain the bolts and other hardware for later reassembly.

**Step 3** Locate an unused option slot connector on the computer motherboard. Loosen the knurled screw securing the blank rear faceplate for this slot, and remove the faceplate.

☞ **Note:** If you remove any other option or change its slot position, you must change the software configuration for the option when you reapply power.

**Step 4** Insert the MC-984 board into the option slot connector. Make sure the board is firmly seated in the connector.

**Step 5** Tighten the knurled screw to secure the board's rear faceplate to the computer frame.

☞ **Note:** This screw is required for proper grounding of the board.

**Step 6** You can reinstall the computer cover now, or wait until you have completed the software configuration. Leaving the cover removed allows you to monitor the board's network indicator as you continue with the configuration.

**Step 7** Attach the Remote I/O drop cable to the RIO F-connector on the MC-984 adapter plate.

**Step 8** With the reference disk installed, reconnect the computer power cable and power up the computer.

**Step 10** Continue with the installation by configuring the controller's Modbus Plus network node address, memory window base address, and interrupt level. Guidelines are provided on the next page.

## 4.6 Configuring the Board

Use these guidelines to configure the board's network node address, memory window base address, and interrupt level, referring to your computer product manuals as required:

**Step 1** If you have just completed the board installation, your computer should be powered up with the reference disk in place. If you have not already done so, insert the reference disk into your computer's disk drive and boot the computer. The computer's 'logo' identification screen should appear.

**Step 2** A message should appear stating that a new option has been installed, with the prompt:

```
Automatically configure the system?  (Y / N)
```

Press `Y` to configure the system.

**Step 3** When the configuration process is complete, The system will prompt you to press <ENTER> to reboot the system. The computer's 'logo' screen should appear again. Press <ENTER> to continue with the configuration.

**Step 4** An initial selections menu will appear. Select menu item 3:

```
Set configuration
```

**Step 5** A configuration menu will appear. Select menu item 2:

```
Change configuration.
```

**Step 6** Use your cursor keys or <Page Up> <Page Down> keys to scroll down to the configuration area. This area displays:

```
MODICON MC984 PROGRAMMABLE CONTROLLER ADAPTER.
```

**Step 7** Use the cursor keys to select the items to be changed for your application. Use F5 <Previous> and F6 <Next> to toggle the item's parameter either downward or upward. Use F1 to get HELP on each entry. The initial configuration parameters are listed below:

```
Enable / Disable Adapter     [ Adapter Enabled ]
```

This parameter should be left Enabled.

```
Adapter Memory Location      [ 0D0000 – 0D07FF ]
```

This parameter is the base address of a 2K bytes (800 hex) memory buffer area used by the board. This address is automatically allocated to avoid conflict with another resource or option. It may therefore be different from the default address shown above. You may leave the address as configured or change it to a new address. Write down the address, as you will need it later when you configure CONFIG.SYS.

```
Link Node Address            [ Node Address 01 ]
```

This parameter should be set to the Modbus Plus network node address that will be used in your application.

```
Adapter Interrupt Level      [ Polled Mode ]
```

This parameter should be left in Polled Mode.

**Step 8** When the configuration entries have been set for your application, press F10 to save the configuration. Then press <ESCAPE> <ESCAPE> <ENTER> to exit the configuration screen and reboot your computer.

**Step 9** Plug the Modbus Plus network cable connector into the board's rear faceplate connector. Tighten the two connector screws to secure the connector.

The Modbus Plus LED should begin flashing a pattern. The pattern will depend upon the status of the MC-984 and other node devices on the network (see Section 4.7).

For example, if another node uses the same address, the indicator pattern will show that a duplicate address has been detected. If no other node is present, the pattern will show a fault condition. Otherwise, if the network is normally active the pattern will show normal operation.

# 4.7 Reading the Network Indicator

The Modbus Plus LED indicator is the leftmost of the three LEDs on the MC-984 board. It shows the communication status at the Modbus Plus port. It can be used to diagnose suspected faults on the network cable. The indicator is accessible with the computer top cover removed.

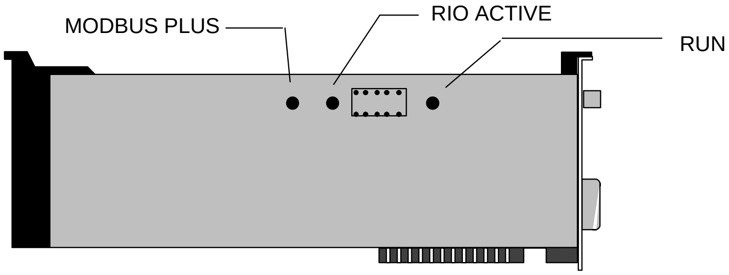


**Figure 27 MC–984 LED Indicator**s

Modbus Plus status is shown by flashing a repetitive pattern on the network indicator. The patterns are:

| LED Pattern | Indication (Status) |
|---|---|
| Six flashes/second | Normal operating state for the node—it is successfully receiving and passing the token. All nodes on a healthy network flash this pattern. |
| One flash/second | The node is off-line just after power-up or after exiting the four flashes/second mode. In this state, the node monitors the network and builds a table of active nodes and token-holding nodes. After being in this state for 5 seconds, the node attempts to go to its normal operating state (indicated by 6 flashes/second). |
| Two flashes, then OFF for two seconds | The node hears the token being passed among the other nodes, but it never receives the token itself—check the network for an open circuit or defective termination. |
| Three flashes, then OFF for 1.7 seconds | The node is not hearing token passing among the other nodes. It periodically claims the token but cannot find another node to which to pass it. Check the network for an open circuit or defective termination. |
| Four flashes, then OFF for 1.4 seconds | The node has heard a valid message from a node using a network address identical to its own address. The node remains in this state for as long as it continues to hear the duplicate address. If the duplicate address is not heard for 5 seconds, the node changes to one flash/second mode. |

### 4.7.1 Network Diagnosis With MBPSTAT

Rather than viewing your network indicator, you may find it more convenient to diagnose suspected faults using your Network Diagnostic Utility program, MBPSTAT.EXE. This utility is supplied on the distribution disk with your controller.

A full description of how to run your MBPSTAT program, and how to select its options for diagnosing your network, is in Appendix D. If you select option 10, ‘Show Node Personality’, your screen will display the same kind of status information that is shown by the flashing patterns of your network indicator. Status is shown in the ‘Peer Status’ line of your MBPSTAT screen.

Here is how your MBPSTAT screen messages correspond to the indicator patterns:

| MBPSTAT Message | Indicator Pattern |
|---|---|
| Normal Link Operation | Six flashes per second |
| Monitor Link Operation | One flash per second |
| Never Getting Token | Two Flashes, then OFF for two seconds |
| Sole Station | Three flashes, then OFF for 1.7 seconds |
| Duplicate Station | Four flashes, then OFF for 1.4 seconds |

## 4.8 Stopped Error Codes

All 984 controllers contain the following stopped error codes, which will be displayed in the panel software:

| Error Code | Mnemonic | Meaning |
|---|---|---|
| 0*x*7FFF | PCSICK | Controller unhealthy |
| 0*x*8000 | PCSTOPPED | Controller stopped |
| 0*x*4000 | BADTCOP | Bad I/O traffic cop table |
| 0*x*2000 | DIMAWAR | PLC in DIM AWARENESS state |
| 0*x*1000 | PORTIVENT | Bad port intervention |
| 0*x*0800 | BADSEGSCH | Bad segment scheduler |
| 0*x*0400 | SONNOTIST | Start of network (SON) did not start segment |
| 0*x*0200 | PDCHEKSUM | Bad power–down checksum |
| 0*x*0080 | NOEOLDOIO | Watchdog timer has expired |
| 0*x*0040 | RTCFAILED | Real time clock failure |
| 0*x*0020 | BADOXUSED | Bad *coil used* table |
| 0*x*0010 | RIOFAILED | Remote I/O failure |
| 0*x*0008 | NODETYPE | Illegal node type used |
| 0*x*0004 | ULCSUMERR | User logic checksum error |
| 0*x*0002 | DSCRDISAB | Discrete disable error |
| 0*x*0001 | BADCONFIG | Bad configuration table |

Stopped error states of various controller nodes may also be sent over the Modbus Plus network using Modbus function 11 hex, as described in Appendix F.

## 4.9 Labeling the Modbus Plus Port

Two sets of labels are provided with your MC-984 Controller card to identify the network and node address at its Modbus Plus port. One label should be attached to the unit when you complete the connection to the network. The other is a spare.

Enter onto the label the Modbus Plus network number and node address you have assigned to the unit's Modbus Plus port. Place the label on the unit so that it can readily identify the port.

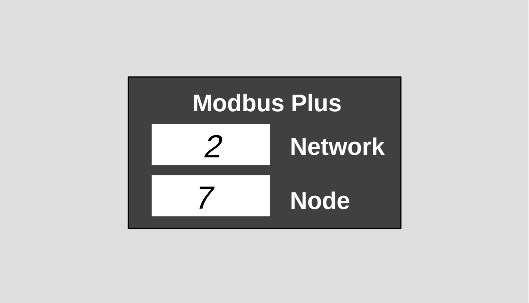


**Figure 28 MC–984 Modbus Plus Port Label**

## 4.10 Initializing the MC–984

During installation of your MC-984, you setup its parameters using a reference disk. These include the network address and memory window base address.

If you recycle power to your host computer, the MC-984 will apply the parameters as they are currently set. If you do a keyboard ‘warm boot’ with Ctrl-Alt-Del, the memory window base address parameter will be affected, but the network address will not. If you want to change the network address, you must recycle power.

During a ‘warm boot’, the device will continue with Modbus Plus activity, including passing tokens and handling any transactions that are active. If it was solving ladder logic, it will continue doing so without interruption.

# 4.11 MC–984 Specifications

**MC–984 Controller Specifications**

| | | |
|---|---|---|
| Description | Name | MC–984 Programmable Controller with Modubs Plus |
| | Part Number | AM–0984–MC0 |
| Physical Characteristics | Size | Standard Single Slot Micro Channel Board,<br>11.5 x 3.475 in (292 x 88 mm) |
| | Weight | 1.0 lb (0.45 kg) net |
| | | 2.0 lbs (0.9 kg) shipping |
| Power | Operating Current | From Computer Motherboard,<br>850 mA typical; 1.2 A maximum |
| Environmental | Temperature | 0 ... 60 degrees C, operating |
| | | –40 ... +80 degrees C, storage |
| | Humidity | 0 ... 95%, non–condensing |
| | EMI, Radiated Susceptibility | MIL STD 461B RS02, RS03 |
| | EMI, Conducted Susceptibility | MIL STD 461B CS02 |
| Network Connection | Connector Type | Mates with Modbus Plus drop cable |
| Software | Operating System | MS–DOS 3.1 or later<br>OS/2 1.0 or later |
| | C Library | Microsoft C 5.1 (large model) |
| | DESQview | Version 2.24 |
| User Memory | Size | 16K Words |
| I/O Capability | RIO Connector | F–Connector for RG6/U RIO Cable |
| | RIO Communication | S908 Protocol |
| | Logic Solve Time | 1.5 ms/K words of user logic |
| | RIO Cable System Loss | Up to 32 dB loss |
| | RIO Drops Supported | Up to 7 drops |
| | Local Drops Supported | None |
| | ASCII Devices Supported (2 per drop) | Up to 14 |

# Chapter 5
# The SM85 Network Adapter

# 5.1 The SM85 and Your Computer

## 5.1.1 Your Hardware Configuration

The SM85 contains no switches or jumpers that are configurable by the user. Its configuration in your host computer is set by software, as outlined below.

Before installing the SM85 board, you'll update the reference disk that is supplied with your host computer. After updating the disk, you can install the board into an available slot in your computer's motherboard, and connect the network cable.

## 5.1.2 Your Software Configuration

The SM85 is supplied with an options file on a disk that will be used to define its presence in your host computer. The file is read using the reference disk that is part of your host computer system. Before installing the SM85, you'll insert your reference disk and follow its menus to copy the option file and update the disk. Then you can install the SM85 board.

You'll use the reference disk again to further define the SM85 configuration. You'll assign its network node address, memory window base address, and other parameters on a software menu. The node address identifies the SM85 for tokens and messages on the Modbus Plus network. The memory address defines a 2K bytes area in your computer that will be used as a buffer between the SM85 and your application.

Next, you can install the SM85 device driver on your hard disk and edit your CONFIG.SYS file to recognize the driver. This will identify the SM85 uniquely to your application, even when multiple options may be present in your computer. Separate drivers are supplied for DOS and OS/2.

You can install the source code, headers, and library files supplied with your SM85. You can compile and link them to your application program.

You also have a network diagnostic utility and a set of sample programs that show methods for accessing controller registers and the network's global database.

Figure 29 summarizes the configuration of the SM85 and your computer product.

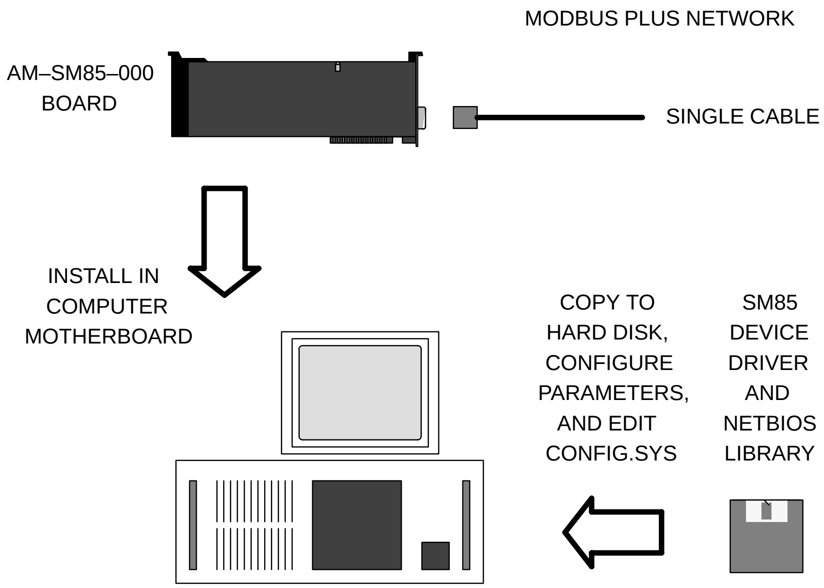

MODBUS PLUS NETWORK
AM–SM85–000 BOARD
SINGLE CABLE
INSTALL IN COMPUTER MOTHERBOARD
COPY TO HARD DISK, CONFIGURE PARAMETERS, AND EDIT CONFIG.SYS
SM85 DEVICE DRIVER AND NETBIOS LIBRARY

**Figure 29 Overview of the SM85 and Host Configuration**

### 5.1.3 System Planning

For further information about planning your Modbus Plus network system, see the *Modibus Plus Network Planning and Installation Guide.*

## 5.2 Installation Overview

Installation of the SM85 Network Adapter consists of four kinds of actions:

□ Updating the computer's reference disk—letting the system automatically copy the SM85 option file to the reference disk (see Section 5.3)

□ Installing the adapter board into the host(see Section 5.4)

□ Configuring the adapter's network address, memory window base address, and interrupt, and connecting it to the network (see Section 5.5)

□ Installing the device driver, software library, network diagnostic, and sample programs. Instructions for installing the DOS and OS/2 versions of your software are provided in separate chapters of this guide.

☞ **Note:** Before installing the SM85, you should be familiar with methods for handling circuit boards, including methods for antistatic protection. If you are not familiar with these methods, contact Modicon for assistance.

### 5.2.1 Adding or Deleting Active Nodes

If you are replacing an SM85 as a node device on an active Modbus Plus network, you can disconnect the device's local drop cable and reconnect it without powering down the devices at the network's other nodes. The network protocol will bypass the removed device, and will include it when reconnected.

**Caution: If your application is dependent upon the presence of the network adapter, adding or deleting it as an active node can produce unpredictable results. Make sure to determine how the application will handle the network configuration before adding or deleting any node.**

If you disconnect a node device from the network, it is not necessary to terminate its local drop connector. The connector should be left open electrically. Cover its pins to prevent damage and contamination.

## 5.3 Updating the Reference Disk

Your computer system includes a reference disk that contains files describing the options that are present. As a part of the SM85 installation, you must update the disk to include the SM85 as a new option. The process is handled by menus when you boot your system.

☞ **Note:** You must update the reference disk before you install the SM85 into your computer product. If you try to start a computer with the SM85 installed without updating the reference disk, the system will return the message: `system error 165`.

You will need a working copy of the reference disk supplied with your system. The disk should not be set for write protection. You will also need disk 1 from the AS-DIBM-220 software package supplied with your SM85. Refer to your computer product manuals as required to do the following steps:

**Step 1** Insert the reference disk into your computer's diskette drive and boot the computer. The computer's 'logo' identification screen should appear.

**Step 2** Press ENTER to clear the logo screen and display the selections menu. Select menu item 5:

```
Copy an option diskette.
```

**Step 3** Following the on-screen instructions, insert your option diskette (AS-DIBM-220, disk 1 of 2), then reinsert the reference disk. The process of updating the reference disk will be handled automatically.

**Step 4** A screen message will appear when the update process is complete. Leave the reference disk in place in the diskette drive. Turn OFF your computer's power switch and continue to the procedure for installing the SM85 board and connecting it to the network cable.

## 5.4 Installing the SM85 Board

Use these guidelines to install the board into your computer product and connect it to the network cable. Refer to your computer product manuals as required to do the following steps.

**Step 1** Set the computer power switch to OFF, and unplug its power cable from the power source.

**Step 2** Remove the computer cover. Retain the bolts and other hardware for later reassembly.

**Step 3** Locate an unused option slot connector on the computer motherboard. Loosen the knurled screw securing the blank rear faceplate for this slot, and remove the faceplate.

☞ **Note:** If you remove any other option, or change its slot position, you will have to change the software configuration for the option when you reapply power.

**Step 4** Insert the SM85 board into the option slot connector. Make sure the board is firmly seated in the connector.

**Step 5** Tighten the knurled screw to secure the board's rear faceplate to the computer frame.

☞ **Note:** This screw is required for proper grounding of the board.

**Step 6** You can reinstall the computer cover now, or wait until you have completed the software configuration. Leaving the cover removed allows you to monitor the network indicator as you continue with the configuration.

**Step 7** With the reference disk installed, reconnect the computer power cable and power up the computer.

**Step 8** Continue to the procedure in the next section for configuring the board.

## 5.5 Configuring the Board

Use these guidelines to configure the board's network node address, memory window base address, and interrupt level, referring to your computer product manuals as required:

**Step 1** If you have just completed the board installation, your computer should be powered up with the reference disk in place. If you have not already done so, insert the reference disk into your computer's disk drive and boot the computer. The computer's 'logo' identification screen should appear.

**Step 2** A message should appear stating that a new option has been installed, with the prompt:

```
Automatically configure the system?  (Y / N)
```

Press **Y** to configure the system.

**Step 3** When the configuration process is complete, The system will prompt you to press <ENTER> to reboot the system. The computer's 'logo' screen should appear again. Press <ENTER> to continue with the configuration.

**Step 4** An initial selections menu will appear. Select menu item 3:

```
Set configuration
```

**Step 5** A configuration menu will appear. Select menu item 2:

```
Change configuration.
```

**Step 6** Use your cursor keys or <Page Up> <Page Down> keys to scroll down to the configuration area. This area displays the title:

```
MODICON SM85 / MODBUS PLUS ADAPTER.
```

**Step 7** Use the cursor keys to select the items to be changed for your application. Use F5 <Previous> and F6 <Next> to toggle the item's parameter either downward or upward. Use F1 to get HELP on each entry. The initial configuration parameters are listed below:

```
Enable / Disable Adapter      [ Adapter Enabled ]
```

This parameter should be left Enabled.

```
Adapter Memory Location       [ 0D0000 – 0D07FF ]
```

This parameter is the base address of a 2K bytes (800 hex) memory buffer area used by the board. This address is automatically allocated to avoid conflict with another resource or option. It may therefore be different from the default address shown above. You may leave the address as configured or change it to a new address. Write down the address, as you will need it later when you configure CONFIG.SYS.

```
Link Node Address             [ Node Address 01 ]
```

This parameter should be set to the Modbus Plus network node address that will be used in your application.

```
Adapter Interrupt Level       [ Polled Mode ]
```

This parameter should be left in Polled Mode.

**Step 8** When the configuration entries have been set for your application, press F10 to save the configuration. Then press <ESCAPE> <ESCAPE> <ENTER> to exit the configuration screen and reboot your computer.

**Step 9** Plug the Modbus Plus network cable connector into the board's rear faceplate connector. Tighten the two connector screws to secure the connector.

The Modbus Plus LED should begin flashing a pattern. The LED pattern will depend upon the status of the SM85 and other nodes on the network (see Section 5.6).

For example, if another node uses the same address, the indicator pattern will show that a duplicate address has been detected. If no other node is present, the pattern will show a fault condition. Otherwise, if the network is normally active the pattern will show normal operation.

## 5.6 Reading the Network Indicator

The board has an indicator that shows the communication status at the Modbus Plus port. It can be used to diagnose suspected faults on the network cable. The indicator is accessible with the computer top cover removed. Figure 30 shows the indicator location.

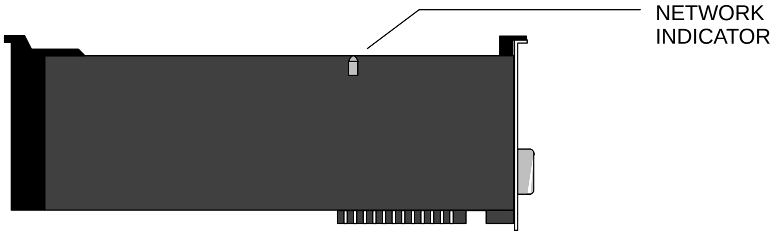


**Figure 30 SM85 Network Indicator**

Modbus Plus status is shown by flashing a repetitive pattern on the network indicator. The patterns are:

| LED Pattern | Indication (Status) |
|---|---|
| Six flashes/second | Normal operating state for the node—it is successfully receiving and passing the token. All nodes on a healthy network flash this pattern. |
| One flash/second | The node is off-line just after power-up or after exiting the four flashes/second mode. In this state, the node monitors the network and builds a table of active nodes and token-holding nodes. After being in this state for 5 seconds, the node attempts to go to its normal operating state (indicated by 6 flashes/second). |
| Two flashes, then OFF for two seconds | The node hears the token being passed among the other nodes, but it never receives the token itself—check the network for an open circuit or defective termination. |
| Three flashes, then OFF for 1.7 seconds | The node is not hearing token passing among the other nodes. It periodically claims the token but cannot find another node to which to pass it. Check the network for an open circuit or defective termination. |
| Four flashes, then OFF for 1.4 seconds | The node has heard a valid message from a node using a network address identical to its own address. The node remains in this state for as long as it continues to hear the duplicate address. If the duplicate address is not heard for 5 seconds, the node changes to one flash/second mode. |

### 5.6.1 Network Diagnosis With MBPSTAT

Rather than viewing your network indicator, you may find it more convenient to diagnose suspected faults using your Network Diagnostic Utility program, MBPSTAT.EXE. This utility is supplied on the distribution disk with your controller.

A full description of how to run your MBPSTAT program, and how to select its options for diagnosing your network, is in Appendix D. If you select option 10, ‘Show Node Personality’, your screen will display the same kind of status information that is shown by the flashing patterns of your network indicator. Status is shown in the ‘Peer Status’ line of your MBPSTAT screen.

Here is how your MBPSTAT screen messages correspond to the indicator patterns:

| MBPSTAT Message | Indicator Pattern |
|---|---|
| Normal Link Operation | Six flashes per second |
| Monitor Link Operation | One flash per second |
| Never Getting Token | Two Flashes, then OFF for two seconds |
| Sole Station | Three flashes, then OFF for 1.7 seconds |
| Duplicate Station | Four flashes, then OFF for 1.4 seconds |

## 5.7 Labeling the Modbus Plus Port

Two sets of labels are provided with your network adapter to identify the network and node address at its Modbus Plus port. One label should be attached to the unit when you complete the connection to the network. The other set is a spare.

Enter onto the label the Modbus Plus network number and node address you have assigned to the unit's Modbus Plus port. Place the label on the unit so that it can readily identify the port. Figure 31 shows an example of the completed label.

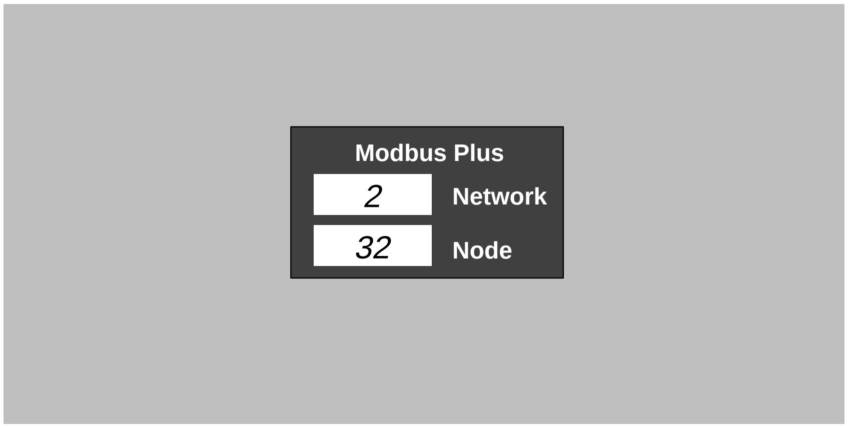


Figure 31 SM85 Modbus Plus Port Label

## 5.8 Initializing the SM85

During installation of your SM85, you setup its parameters using a reference disk. These include the network address and memory window base address.

If you recycle power to your host computer, the SM85 will apply the parameters as they are currently set. If you do a keyboard ‘warm boot’ with Ctrl-Alt-Del, the memory window base address parameter will be affected, but the network address will not. If you want to change the network address, you must recycle power.

During a ‘warm boot’, the device will continue with Modbus Plus activity, including passing tokens and handling any transactions that are active.

# 5.9 SM85 Specifications

**SM85 Network Adapter Specifications**

| | | |
|---|---|---|
| Description | Name | SM85 Modbus Plus Adapter |
| | Part Number | AM–SM85–000 |
| Physical Characteristics | Size | Standard Micro Channel Board, 11.5 x 3.475 in (292 x 88 mm) |
| | Weight | 1.0 lb (0.45 kg) net |
| | | 2.0 lbs (0.9 kg) shipping |
| Power | Operating Current | From Computer Motherboard, 750 mA at 5 Vdc maximum |
| Environmental | Temperature | 0 ... 60 degrees C, operating |
| | | –40 ... +80 degrees C, storage |
| | Humidity | 0 ... 95%, non–condensing |
| | EMI, Radiated Susceptibility | MIL STD 461B RS02, RS03 |
| | EMI, Conducted Susceptibility | MIL STD 461B CS02 |
| Network Connection | Connector Type | Mates with Modbus Plus drop cable |
| Software | Operating System | MS–DOS 3.1 or later<br>OS/2 1.0 or later |
| | C Library | Microsoft C 5.1 (large model) |
| | DESQview | Version 2.24 |

# Chapter 6
# Installing the DOS Files

## 6.1 Installing the Device Driver

Before installing your files, review the information file README.DOC on your disks. This file lists each of your programs and describes any recent updates. Installation of the device driver is a three-step process:

□ Copying the device driver file to the host computer's hard disk

□ Editing the host computer's CONFIG.SYS file

□ Initializing the host computer to recognize the new CONFIG.SYS configuration.

### 6.1.1 Copying MBPHOST.SYS

MBPHOST.SYS is the DOS driver file for the SA85, SM85, AT-984, and MC-984.

The driver file is provided on 5.25 in (360 Kbyte) and 3.5 in (720 Kbyte) diskettes. Use the DOS `COPY` command to copy the driver file to your hard disk. Copy the file into a directory that will be recognized by the path that you use in your CONFIG.SYS file (see Section 6.1.2). Here is an example for copying the file to the C: drive root directory:

```
COPY MBPHOST.SYS C:\
```

### 6.1.2 Editing CONFIG.SYS

Edit the CONFIG.SYS file in your host PC to include a unique DEVICE command for each Modicon host based device installed. Each DEVICE command must include the path to the directory containing the device driver file.

The format for the command line is:

```
DEVICE=MBPHOST.SYS /Mnnnn /Nn /Snn /R2 /B
```

| Switch | Meaning | Example | Default |
|---|---|---|---|
| /Mnnnn | Selects the memory window base address | /MD080 = address D0800 | D0000 |
| /Nn | Selects a specific device within the host | /N1 = device 1 | device 0 |
| /Snn | Selects the software interrupt vector | /S5B = interrupt 5B | 5C |
| /R2 | Defines driver revision 2.0 | /R2 (required entry) | none |
| /B | Blocks Board Reset and SA85OFF commands | /B = do not allow commands | allowed |

**/M**
Use this argument to specify the same address that was set with the board's memory window base address DIP switches (SA85, AT-984) or software setup configuration (SM85, MC-984). Only four hexadecimal digits are used after the `/M` argument (e.g., `/MD080`) -- the rightmost 0 in the address is assumed.

**/N**
Use this argument to specify a unique Modicon device number within the host. Up to two devices can installed in the host, numbered 0 and 1. This argument identifies each board to the ncb_open() function in NetBIOS. If a value is not specified, the device driver defaults to device number 0.

**/S**
Use this argument to relocate a Modicon device's software interrupt, avoiding conflict with other interrupt sources. The valid range is 00 ... FF hexadecimal. If multiple Modicon devices are installed in a single host, they should all use interrupt 5C, the standard NetBIOS interrupt.

**Note:** Interrupt 5C is the default for running the network diagnostic and sample programs supplied with your devices. These programs accept a command argument for using a different interrupt (see the appendixes in this guide for those programs).

**/R2**
This argument is required for normal operation of the board. It specifies that Revision 2.0 of the driver is installed.

**/B**
Use this argument to prevent Board Reset or SA85OFF commands in your application from resetting the board. If the command is not used, these commands will cause the board to initialize and start running its diagnostics.

### 6.1.3 Initializing the Host Computer

After you have installed the required device drivers and edited CONFIG.SYS, simultaneously press <CTRL> <ALT> <DEL> to initialize the host computer. This will allow the host to recognize the CONFIG.SYS file changes.

If the host based device's switch settings, software setup, or device driver are incorrect, the host computer may hang. If this happens, recheck the installation.

## 6.2 Installing the Remaining Files

### 6.2.1 Installing the MBPSTAT and \TESTSRC Programs

The Network Diagnostic Utility program MBPSTAT.EXE is contained on your disk. You also have a set of test programs (source code and executables) in \TESTSRC that illustrate how to access controller registers and the network's global data.

Copy the diagnostic and test program files to the directory you will want to use for exercising, testing, and debugging your Modbus Plus networking application. Descriptions of the programs are provided in the appendixes of this guide.

### 6.2.2 Installing the \MBPHINC and \NETLIB Files

The two header files NETLIB.H and NETBIOS.H are in \MBPHINC on your disk. The library file NETLIB.LIB is contained in \NETLIB. The Modbus Plus dynamic link library file MBPLUS.DLL, its import library file MBPLUS.LIB, and several .ASM and .MAK files are also contained in \NETLIB.

Refer to the manuals for the compiler you will use for your application program. Copy the contents of \MBPHINC header files to your compiler's 'include' directory. Copy the contents of \NETLIB to your 'library' directory.

NETLIB.LIB was compiled with Borland C version 3.1, using the Large model. A compatible makefile is provided for Microsoft C version 5.1. DESQV.ASM is provided for use with DESQVIEW. You can link these files as required into your application program.

### 6.2.3 What to Do Next

After completing the software installation, your host based device is ready to be accessed through NetBIOS functions that you can program in your application. Descriptions of the NetBIOS functions, header files, and library are provided in the appendixes of this guide.

# Chapter 7
# Installing the OS/2 Files

□ Installing the Driver and Link Files

□ Installing the Remaining Files

## 7.1 Installing the Driver and Link Files

Before installing your files, review the information file READ.ME on your disks. This file lists each of your software programs and describes any recent updates. The text file MPHOS2.DOC contains a detailed description of your driver.

Installation of the device driver and dynamic link library is a three-step process:

- □ Copying the driver and dynamic link library files to the host computer's hard disk
- □ Editing the host computer's CONFIG.SYS file
- □ Initializing the host computer to recognize the new CONFIG.SYS configuration.

### 7.1.1 Copying MPHOST.SYS, MPHTICK.EXE, and SA85.DLL

MPHOST.SYS is the OS/2 driver for the single-cable or dual-cable SA85, SM85, AT-984, and MC-984. MPHTICK.EXE is the OS/2 driver application program. SA85.DLL is the Modicon NetBIOS dynamic link library file.

Use the OS/2 `COPY` command to copy the driver and dynamic link library files to your hard disk. Here are examples for copying the files to the C: OS/2 directories:

```
COPY MPHOST.SYS C:\OS2
COPY MPHTICK.EXE C:\OS2
COPY SA85.DLL C:\OS2\DLL
```

### 7.1.2 Editing CONFIG.SYS

Use your OS/2 system editor to add a DEVICE and RUN line for each installed device. Modify the LIBPATH environment to contain the path of the SA85.DLL file. A sample CONFIG.SYS file is provided on your disk. Here are examples:

```
DEVICE=C:\OS2\MPHOST.SYS /MC000 /N1
LIBPATH=C:\OS2\DLL
RUN=C:\OS2\MPHTICK.EXE /N1
```

The following parameters should also be present in CONFIG.SYS:

```
threads=128
memman=SWAP,MOVE
maxwait=3
timeslice=32,100
```

The DEVICE command line options are:

| Switch | Meaning | Example | Default |
|---|---|---|---|
| /Mnnnn | Selects the memory window base address | /MD080 = address D0800 | D0000 |
| /Nn | Selects a specific device within the host | /N1 = device 1 | device 0 |
| /B | Blocks Board Reset and SA85OFF commands | /B = do not allow commands | allowed |

**/M**
Use this argument to specify the same address that was set with the board's memory window base address DIP switches (SA85, AT-984) or software setup configuration (SM85, MC-984). Only four hexadecimal digits are used after the /M argument (e.g., /MD080) -- the rightmost 0 in the address is assumed.

**/N**
Use this argument to specify a unique Modicon device number within the host. Up to two devices can installed in the host, numbered 0 and 1. This argument identifies each board to the ncb_open() function in NetBIOS. If a value is not specified, the device driver defaults to device number 0.

**/B**
Use this argument to prevent Board Reset or SA85OFF commands in your application from resetting the board. If the command is not used, these commands will cause the board to initialize and start running its diagnostics.

### 7.1.3 Initializing the Host Computer

After you have installed the required device drivers and edited CONFIG.SYS, select 'Shutdown . . .' from the desktop window's 'Desktop' menu. Follow the screen instructions for shutdown. This will allow the host to recognize the CONFIG.SYS file changes.

If the host based device's switch settings, software setup, or device driver are incorrect, the host computer may hang. If this happens, recheck the installation.

# 7.2 Installing the Remaining Files

## 7.2.1 Installing the Diagnostic and Sample Programs

The Network Diagnostic Utility program MBPSTAT.EXE is contained on your disk. You also have a set of sample programs that illustrate how to access controller registers and the network's global data.

Copy the diagnostic and sample program files to the directory you will want to use for exercising, testing, and debugging your Modbus Plus networking application. Descriptions of the programs are provided in the appendixes of this guide.

## 7.2.2 Installing the Headers, Source, and Library Files

The header files NETLIB.H and NETBIOS.H, source code file NETLIB.C (with your NetBIOS functions), and library file NETLIB.LIB are contained on your disk.

Refer to the manuals for the compiler you will use for your application program. Copy NETLIB.H and NETBIOS.H to your compiler's 'include' directory (typically \INCLUDE). Copy NETLIB.C to your compiler's working directory (typically \BIN). Copy NETLIB.LIB to the 'library' directory (typically \LIB).

NETLIB.LIB was compiled with Borland C version 3.1, using the Large model. You can link this file as required into your application program.

## 7.2.3 What to Do Next

After completing the software installation, your host based device is ready to be accessed through NetBIOS functions that you can program in your application. Descriptions of the NetBIOS functions, header files, and library are provided in the appendixes of this guide.

# Appendix A.
# Using NetBIOS Functions

# A.1 The Modbus Plus NetBIOS Environment

NetBIOS is a C language software interface developed by IBM for PC networking. Modicon has adapted many of the NetBIOS functions to enable them to be used directly and efficiently in Modbus Plus networking. The applications designer who is familiar with standard NetBIOS functions will find the Modicon implementation easy to understand and apply.

Functions are provided for: initializing the local host based device; opening paths between the device and remote nodes; sending and receiving data or global data; receiving statistics from remote nodes; and retrieving the local device's statistics. The host can construct Modbus messages in buffers and send them to nodes. Incoming Modbus messages can be stored for use in the host application.

The syntax, arguments, and return values for these functions are described in this appendix. When using these functions, the designer should note their differences from the standard NetBIOS functions.

Before coding your application, you might want to examine the source code file netlib.c to view the NetBIOS function definitions. This file is on your disk. Also, you'll find a set of sample programs on your disk. You can examine them to see the use of these functions in actual ModbusPlus applications. You might want to printout a hard copy of the files to use as a reference in coding your application.

## A.1.1 The Network Control Block

The data structure manipulated by the NetBIOS commands is the Network Control Block (NCB). As a NetBIOS command executes, all data is passed in the NCB, including pointers to external buffers.

The NCB is 64 bytes long and contains 14 fields of information. The fields are described on the next page. The columns are:

□ Offset—the address of the field within the NCB (in hexadecimal)

□ Size—the size in bytes of the field (in decimal)

□ Field Name—the variable name of the field

□ Field Description—the type of function or service performed by the field.

NCB fields are defined in the header file netbios.h. A source code listing of the file is provided in Appendix C.

| Offset<br>(Hex) | Size<br>(Bytes) | Field Name | Field Description |
|---|---|---|---|
| 0 | 1 | NCB_COMMAND | NetBIOS command code—high order bit set indicates no-wait (see netbios.h). |
| 1 | 1 | NCB_RETCODE | Completion result. - A value of zero if no error. See the list of return codes in this Appendix. |
| 2 | 1 | NCB_LSN | Not used and not supported. |
| 3 | 1 | NCB_NUM | A value identifying the type of internal path being used by the command. The value is placed in the field when the NCB is opened:<br>0x01 ... 0x08 DM1 ... DM8<br>0x41 ... 0x48 DS1 ... DS8<br>0x81 ... 0x88: PM1 ... PM8<br>0xC1 ... 0xC8 PS1 ... PS8 |
| 4 | 4 | NCB_BUFFER | Address of the message buffer |
| 8 | 2 | NCB_LENGTH | Length of the message buffer, in bytes |
| 0A | 16 | NCB_CALLNAME | The first 7 bytes identify the Modbus Plus routing path being used by the command. The contents are placed in the field when the NCB is opened. The contents are:<br>Bytes 0 and 1 'DM', 'DS', 'PM', or 'PS'<br>Bytes 2 ... 6 Routingaddresses(binary)<br>Other bytes are not used and not supported. |
| 1A | 16 | NCB_NAME | The user's name on the network |
| 2A | 1 | NCB_RTO | Receive timeout value in 500 ms increments. Used with ncb_receive_wait() and ncb_receive_datagram() |
| 2B | 1 | NCB_STO | Send timeout value in 500 ms increments. Used with ncb_send() and ncb_send_datagram() |
| 2C | 4 | NCB_POST_ADDRESS | Not used and not supported |
| 30 | 1 | NCB_LANA_NUM | Modbus Plus adapter number—must be 0 or 1. |
| 31 | 1 | NCB_CMD_CPLT | Not used and not supported |
| 32 | 14 | NCB_RESERVE | Reserved field, used internally by NetBIOS |

## A.2 A Summary of Commands

The Modbus Plus NetBIOS commands and their corresponding functions in the NETLIB library are listed below.

Each function except exec_netbios() first initializes fields of the NCB structure. The function then calls exec_netbios() which invokes the int86x interrupt.

| Command | NETLIB Function | Purpose |
|---|---|---|
| RESET | ncb_reset() | Resets the host based device and runs diagnostics |
| – – | ncb_sa85off() | Turns off the device driver |
| CANCEL | ncb_cancel() | Aborts a transaction |
| STATUS | ncb_status() | Returns adapter status |
| ADD_NAME | ncb_open() | Obtains a path |
| DELETE_NAME | ncb_close() | Releases a path |
| SEND | ncb_send() | Sends a packet |
| RECEIVE | ncb_receive() | Receives a packet, no wait for response |
| RECEIVE_WAIT | ncb_receive_wait() | Same as RECEIVE except that it waits until the response is received or a timeout occurs |
| SEND_DATAGRAM | ncb_send_datagram() | Sends global data |
| RECEIVE_DATAGRAM | ncb_receive_datagram() | Receives global data |
| SET_PEER_PRM | ncb_set_peer_params() | Sets parameters for peer cop configuration |
| XFER_GLBINP | ncb_xfer_glob_inp() | Transfers all newly received peer–copped global input data from the network to the user's buffer |
| XFER_SPCOUT | ncb_xfer_spec_out() | Transfers all peer–copped specific output data from the user's buffer to the network |
| XFER_SPCINP | ncb_xfer_spec_inp() | Transfers all newly received peer–copped specific input data from the network to the user's buffer |
| PUT_PEERCOP | ncb_put_peer_cop() | Transfers the current peer cop configuration to the peer processor |
| – – | ncb_set_slave_login() | Sets the login status on a slave path |
| – – | ncb_set_sw_interrupt() | Sets the software interrupt vector |
| – – | exec_netbios() | Executes a NetBIOS command through the int86x interrupt. This function is an entry point for the device driver. Do not use it in your application. |

Note that Peer Cop commands apply to the model AM-SA85-002 host-based adapter only.

### A.2.1 Path Requirements

Some functions require the existence of a path in the local host-based adapter before they can execute. Four types of paths are available: DM (Data Master), DS (Data Slave), PM (Program Master), and PS (Program Slave). Paths are defined and established by the ncb_open() function, and are released by ncb_close(). Other functions do not require a path and can be executed without one. This Appendix shows the types of paths required by each function.

### A.2.2 Message Buffers

The host application passes a buffer pointer value to ncb_send(), ncb_receive(), or ncb_receive_wait() that points to a buffer for the data to be passed in the function. Buffers can be constructed with messages in Modbus format for transactions with Modicon PLCs.

A pointer is passed to ncb_send_datagram() or ncb_receive_datagram() to identify the buffer for global data to be passed in the function.

### A.2.3 Function Definitions and Prototypes

The function definitions for the NETLIB library are contained in the file netlib.c. You may want to print a hard copy of this file to use as a reference in coding your application.

The function prototypes are in the header file netlib.h.

### A.2.4 NCB Structure Definition and NetBIOS Command Codes

The NCB structure is defined in the header file netbios.h. This file also defines the NetBIOS command codes.

### A.2.5 Function Return Codes

The NCB_RETCODE values are defined in the header file netbios.h.
A description of return code values is provided on the following pages.

## A.3 Return Codes

A hex value is returned to the NCB_RETCODE field to show the completion status of a function. The header file netbios.h defines the codes.

| Code | include Variable | Description |
|---|---|---|
| 00 | `#define ERR_success 0` | Normal completion—no error |
| 01 | `#define ERR_bad_buffer_length 1` | Illegal send or status buffer length—send command length > 512 bytes, or status command length < the minimum allowed |
| 03 | `#define ERR_invalid 3` | Invalid NetBIOS command |
| 05 | `#define ERR_timeout 5` | The timeout interval specified for a data send or receive command has expired; or an internal timer for a status command has expired |
| 06 | `#define ERR_buffer_too_small 6` | Receive buffer too small—the buffer size specified in NCB_LENGTH is not large enough to hold the receive data |
| 08 | `#define ERR_bad_session_num 8` | Invalid local session number in NCB_LSN |
| 09 | `#define ERR_no_RAM 9` | Out of resources – the network adapter does not have enough RAM to process the command |
| 0A | `#define ERR_session_closed 0xa` | Session has been terminated by the remote node |
| 0B | `#define ERR_cancel 0xb` | Command cancelled—execution of NCB commands aborted by the ncb_cancel() command |
| 0D | `#define ERR_dup_local_name 0xd` | Duplicate local name—an ncb_open() command has specified a name that already exists |
| 0E | `#define ERR_name_table_full 0xe` | The local name table is full |
| 0F | `#define ERR_active_session 0xf` | An ncb_close() command has been executed, but the name has active transactions; the name will not be deleted until all its sessions are closed |
| 11 | `#define ERR_sess_table_full 0x11` | The local session table is full |
| 12 | `#define ERR_no_listen 0x12` | Routing failure; remote node is not responding to the transaction |
| 13 | `#define ERR_bad_name_num 0x13` | Bad value in NCB_NUM field |
| 14 | `#define ERR_no_answer 0x14` | No answer to CALL, or no such remote |
| 15 | `#define ERR_no_local_name 0x15` | Invalid name entry in local name table |
| 16 | `#define ERR_duplicate_name 0x16` | Name already exists elsewhere on the network |
| 17 | `#define ERR_bad_delete 0x17` | Name has been incorrectly deleted |
| 18 | `#define ERR_abnormal_end 0x18` | Session terminated abnormally—connection to remote node terminated |
| 19 | `#define ERR_name_error 0x19` | Name conflict—two nodes using the same name have been detected |
| 1A | `#define ERR_bad_packet 0x1a` | Bad NetBIOS packet on network |
| 21 | `#define ERR_card_busy 0x21` | Adapter busy—command cannot execute |
| 22 | `#define ERR_too_many_cmds 0x22` | Too many commands queued |
| 23 | `#define ERR_bad_card_num 0x23` | Invalid value for NCB_LANA_NUM—must be 0 or 1 |

|  |  |  |
|---|---|---|
| 24 | #define ERR_cancel_done 0x24 | Command completed before ncb_cancel() executed; code returned in ncb_cancel() when previous command completed normally |
| 26 | #define ERR_no_cancel 0x26 | Invalid ncb_cancel() command—target NCB cannot be found |
| 30<br>...<br>3D | #define ERR_invalid 0x30<br>...<br>#define ERR_invalid 0x3D | Invalid NetBIOS command |
| 40<br>...<br>FE | (not applicable to #define) | Hardware error |
| FF | #define ERR_busy 0xff | Still processing command |

## A.4 ncb_cancel( )

### NetBIOS Command

CANCEL

### Description

Cancels and aborts a transaction before it has completed. The function does not clear an existing path—if the path is to be closed, issue a call to ncb_close() after completion of ncb_cancel().

### Path Required

Requires the existence of a DM, DS, PM, or PS path.

### Syntax

```
int ncb_cancel(*ncbp)
NCB *ncbp ;
```

| Argument | |
|---|---|
| ncbp | A pointer to the NCB to be canceled. |

| Return Value | |
|---|---|
| ERR_success | Always |

# A.5 ncb_close( )

### NetBIOS Command

DELETE_NAME

### Description

Closes a transaction path that was opened with ncb_open(). If a transaction is pending on the path, the transaction is aborted.

### Path Required

Requires the existence of a DM, DS, PM, or PS path.

### Syntax

```
int ncb_close(*ncbp)
NCB *ncbp ;
```

| Argument | |
|---|---|
| ncbp | A pointer to the NCB returned by the previous ncb_open(). |

| Return Values | |
|---|---|
| ERR_success | Normal completion |
| ERR_bad_name_num | Invalid path |

## A.6 ncb_open( )

### NetBIOS Command

ADD_NAME

### Description

Opens a transaction path in the local host based device and designates a node with which you wish to communicate. You must call this function before you can communicate with another node. Pass an argument to the function that points to a string containing the path type and routing information. The path types are:

□ DM—Data master path

□ PM—Program master path

□ DS—Data slave path

□ PS—Program slave path

The string format is:

“PP.R1.R2.R3.R4.R5”

where:

□ PP is the path type in the host based device: DM, PM, DS, or PS

□ R1 ... R5 represent each level of Modbus Plus routing to the destination node

A period ( . ) character separates the fields within the string. You must zero-fill unused fields in the string, as in the examples below.

### Example

Modbus Plus routing paths are described fully in Chapter 1. Here are two examples:

□ Routing to a network adapter: DM.9.2.0.0.0 opens a Data Master path to task (slave path) 2 in the network adapter at node address 9.

□ Routing to a controller: DM.8.3.0.0.0 opens a Data Master path to the controller at node 3 on the network that is accessed through the Bridge Plus device at node 8 on the local network.

### Sample Call

```
   ncb_open(”DM.8.3.0.0.0”, 0) ;
```

where

□ `”DM.8.3.0.0.0”` is a string defining the path type and routing.

□ `0` selects host based device 0.

### Pointer to NCB

The function returns a pointer to an NCB, or a NULL on an error. The pointer must be passed to the following commands:

| | | |
|---|---|---|
| ncb_send() | ncb_receive_wait() | ncb_close() |
| ncb_receive() | ncb_set_slave_login() | ncb_cancel() |

### Path Required

Not applicable. The function establishes a new DM, DS, PM, or PS path.

### Syntax

```
NCB *ncb_open(*name, lan)
char *name ;
int lan ;
```

**Arguments**

| | |
|---|---|
| name | A pointer to a string containing the path type and routing. |
| lan | Selects the host based device (0 or 1). |

**Return Values**

| | |
|---|---|
| pointer to an NCB | See **Pointer to NCB** above |
| NULL | If an error has occurred |

# A.7 ncb_put_peer_cop( )

### NetBIOS Command

PUT_PEERCOP

### Description

Transfers the peer cop data transfer configuration to the peer processor. This consists of the amounts of words to be transferred for all possible peer copped communications. A transfer length of zero indicates that there is no communication of that type with the associated node.

The host application must first be set into its Run state by issuing ncb_set_peer_params() and ncb_status(), before this command will be accepted by the peer processor. The normal sequence is:

ncb_set_peer_params()
ncb_status()
ncb_put_peer_cop().

**Word Count Limits:** Up to 32 words of global output may be configured for the target node. Global input, specific input, and specific output may each have a total of 500 words.

The command always transfers global and specific data transfer lengths for 64 node addresses. If a node address is not currently configured on the network, its transfer lengths should be specified as zero.

### Path Required

Not applicable. The function does not require a path.

### Syntax

```
int far ncb_put_peer_cop (NCB far * ncbp, int adaptno, char far*
buffer)
NCBP far* ncbp;
int adaptno;
char far *buffer;
```

**Arguments**

| | |
|---|---|
| ncbp | A pointer to the NCB returned by the previous ncb_open(). |
| adaptno | The host device adapter number – must be 0 or 1 |
| buffer | A pointer to the user buffer (see Buffer Format) |

**Return Values**

| | |
|---|---|
| ERR_success | Normal completion |
| ERR_invalid | Data not transferred |

### Buffer Format

Command buffer:
byte 0: global output transfer length
byte 1: node 1 global input transfer length
byte 2: node 2 global input transfer length
. . .
byte 64: node 64 global input transfer length
byte 65: node 1 specific output transfer length
byte 66: node 2 specific output transfer length
. . .
byte 128: node 64 specific output transfer length
byte 129: node 1 specific input transfer length
byte 130: node 2 specific input transfer length
. . .
byte 192: node 64 specific input transfer length
byte 193: peer cop health timeout, 1 ... 100

Response buffer:  Not applicable.

# A.8 ncb_receive( )

### NetBIOS Command

RECEIVE

### Description

Receives a message on the specified path. If the response is available, the function returns with the response. If the response is not available, it returns immediately with ERR_timeout status.

Use this function instead of ncb_receive_wait() to enable better throughput in a multitasking environment.

### Path Required

Requires the existence of a DM, DS, PM, or PS path.

### Syntax

```
int ncb_receive(*ncbp, *buffer)
NCB *ncbp ;
char *buffer ;
```

**Arguments**

| | |
|---|---|
| ncbp | A pointer to the NCB returned by the previous ncb_open(). |
| buffer | A a pointer to the buffer for the received message. |

**Return Values**

| | |
|---|---|
| ERR_success | Good |
| ERR_bad_session_num | NCB_NUM == 0 |
| ERR_bad_name_num | Invalid path |
| ERR_timeout | No response available |
| ERR_invalid | No response pending |
| ERR_busy | Bad path (NCB_NUM) |
| ERR_invalid | Returned length of 0 (NCB_LENGTH) |
| ERR_no_listen | Node not present |

# A.9 ncb_receive_datagram( )

### NetBIOS Command

RECEIVE_DATAGRAM

### Description

Receives a specified quantity of global data into a buffer. The user must allocate an NCB for the function that is separate from NCBs allocated by ncb_open(). After the return, NCB_LENGTH contains the count of global data words received.

### Path Required

Not applicable. The function does not require a path.

### Syntax

```
int ncb_receive_datagram(*ncbp, node, *buffer, timeout, adaptno)
NCB *ncbp ;
int node ;
char *buffer ;
unsigned char timeout ;
int adaptno ;
```

**Arguments**

| | |
|---|---|
| ncbp | A pointer to user–allocated NCB (should be empty). |
| node | The node from which to receive global data (range: 1 ... 64). |
| buffer | A pointer to the buffer that will contain the received data. |
| timeout | The timeout value in units of 0.5 s. |
| adaptno | The host device adapter number—must be 0 or 1. |

**Return Values**

| | |
|---|---|
| ERR_success | Good |
| ERR_card_busy | Node not found |
| ERR_no_answer | No global data present |

# A.10 ncb_receive_wait( )

### NetBIOS Command

RECEIVE_WAIT

### Description

Receives a message on the specified path. Unlike ncb_receive, this function waits until the response comes in or until the timeout occurs before it returns.

### Path Required

Requires the existence of a DM, DS, PM, or PS path.

### Syntax

```
int ncb_receive_wait(*ncbp, *buffer, timeout)
NCB *ncbp ;
char *buffer ;
unsigned char timeout ;
```

**Arguments**

| | |
|---|---|
| `ncbp` | A pointer to the NCB returned by the previous ncb_open(). |
| `buffer` | A a pointer to the buffer for the received message. |
| `timeout` | The timeout value in units of 0.5 s. |

**Return Values**

| | |
|---|---|
| `ERR_success` | Good |
| `ERR_bad_session_num` | NCB_NUM == 0 |
| `ERR_bad_name_num` | Invalid path |
| `ERR_timeout` | No response available |
| `ERR_invalid` | No response pending |
| `ERR_busy` | Bad path (NCB_NUM) |
| `ERR_invalid` | Returned length of 0 (NCB_LENGTH) |
| `ERR_no_listen` | Node not present |

# A.11 ncb_reset( )

### NetBIOS Command

RESET

### Description

Resets the network adapter, runs its diagnostics, and returns it to a known initial state. The function may require up to 100 ms to execute.

This function can be either allowed or blocked by the device driver through the use of driver command line parameters.

The function will be blocked if you are running under Microsoft Windows 3.x or under DesqVIEW. If the function is called, it will have no effect unless the driver has crashed and requires a reset.

### Path Required

Not applicable. The function does not require a path.

### Syntax

```
int ncb_reset(adaptno)
int adaptno ;
```

| Argument | |
|---|---|
| adaptno | The number of the adapter to be reset—must be either 0 or 1. |

| Return Values | |
|---|---|
| ERR_success | Reset occurred normally |
| ERR_invalid | Bad adapter number |
| 0*x*80 | Diagnostic failed |

## A.12 ncb_sa85off( )

### NetBIOS Command

None

### Description

Turns off the device driver. Use this function when your application no longer needs the host based device, and you want to run some other type of program such as a terminal emulation program. It turns off the polling of the host based device by the device driver. Use the ncb_reset() function to reset the device and restore its operation.

This function can be either allowed or blocked by the host based device driver through the use of driver command line parameters.

The function will be blocked if you are running under Microsoft Windows 3.x or under DesqVIEW. If the function is called, it will have no effect.

### Path Required

Not applicable. The function does not require a path.

### Syntax

```
int ncb_sa85off(adaptno)
int adaptno ;
```

**Argument**

| | |
|---|---|
| `adaptno` | The number of the adapter to turn off. Must be either 0 or 1. |

**Return Values**

| | |
|---|---|
| `ERR_success` | Normal completion |
| `ERR_bad_card_num` | Bad adapter number |

# A.13 ncb_send( )

### NetBIOS Command

SEND

### Description

Sends a message on the selected path. For a master path, you must issue ncb_send() before ncb_receive() or an error will be returned. For a slave path, you must issue ncb_receive() before ncb_send() or an error is returned.

### Path Required

Requires the existence of a DM, DS, PM, or PS path.

### Syntax

```
int ncb_send(*ncbp, length, *buffer, timeout)
NCB *ncbp ;
int length ;
char *buffer ;
unsigned char timeout ;
```

**Arguments**

| | |
|---|---|
| ncbp | A pointer to the NCB returned by the previous ncb_open(). |
| length | The length of the message buffer contents in bytes. |
| buffer | A pointer to the buffer containing the message to be sent. |
| timeout | The timeout value in units of 0.5 s. |

**Return Values**

| | |
|---|---|
| ERR_success | Good |
| ERR_bad_session_num | NCB_NUM == 0 |
| ERR_invalid | If response is still pending on the master path, or if no previous receive has been issued on the slave path |

## A.14 ncb_send_datagram( )

### NetBIOS Command

SEND_DATAGRAM

### Description

Sends a message containing a specified quantity of global data. The user must allocate an NCB for the function that is separate from NCBs allocated by ncb_open().

### Path Required

Not applicable. The function does not require a path.

### Syntax

```
int ncb_send_datagram(*ncbp, length, *buffer, timeout, adaptno)
NCB *ncbp ;
int length ;
char *buffer ;
unsigned char timeout ;
int adaptno ;
```

**Arguments**

| | |
|---|---|
| ncbp | A pointer to user–allocated NCB (should be empty). |
| length | The length of the message buffer contents in bytes. |
| buffer | A pointer to the buffer containing the message to be sent. |
| timeout | The timeout value in units of 0.5 s. |
| adaptno | The host device adapter number—must be 0 or 1. |

**Return Values**

| | |
|---|---|
| ERR_success | Good |
| ERR_card_busy | Adapter not found or is initialized |
| ERR_bad_card_num | Bad adapter number |
| ERR_bad_buffer_length | Buffer length > 64 |

# A.15 ncb_set_peer_params( )

### NetBIOS Command

SET_PEER_PRM

### Description

Assigns parameters for the adapter's peer cop configuration. If this function is not called in the user program, the default values are those shown in buffer[0] ... buffer[4] in the Buffer Format section.

This function must be followed by the ncb_status() function to set the application's Run state and timeout value. The normal sequence is:

ncb_set_peer_params()
ncb_status()
ncb_put_peer_cop().

### Path Required

Not applicable. The function does not require a path.

### Syntax

```
int far ncb_set_peer_params(*ncbp, adaptno, *buffer)
NCB far*ncbp ;
int adaptno ;
char far *buffer ;
```

**Arguments**

| | |
|---|---|
| ncbp | A pointer to the NCB returned by the previous ncb_open(). |
| adaptno | The host adapter device number—must be either 0 or 1. |
| buffer | A pointer to the buffer containing the parameters to be set (see Buffer Format for details):<br>Byte 0: controls Statistics Counters.<br>Byte 1: sets Application Run status.<br>Byte 2: specifies origin of Application Run status.<br>Byte 3: specifies method of initializing of Diagnostic Timer.<br>Byte 4: multiplier value for Diagnostic Timer. |

**Return Values**

| | |
|---|---|
| ERR_success | Parameters set successfully |

### Buffer Format

| Byte | Default | Options |
|---|---|---|
| 0 | 0 | 0: Do not clear Statistics Counters.<br>1: Clear counters. |
| 1 | 1 | 1: Host application is set to Running (Note 1, 2, 3).<br>0: Host application is set to Not Running. |
| 2 | 1 | 1: Application Run status is provided in buffer[1].<br>0: Application Run status is provided by hardware. |
| 3 | 1 | 1: Peer Processor Timer is initialized by the driver, or by any interface command. The timer uses a fixed timeout value of 2.55 seconds.<br>0: Peer Processor Timer is initialized by the driver, or by the ncb_set_peer_params() command only. The timer uses a timeout value of 2.55 seconds times the multiplier value specified in buffer[4]. |
| 4 | 0 | Peer Processor Timer multiplier value, increments of 2.55 seconds. Allowable values are:<br>0: Infinite (never times out)<br>1 ... 255: increments of 2.55 seconds. |

### Notes

1. Following the ncb_set_peer_params() command, the host application must issue an ncb_status() command to set it to the Running or Not Running mode.

2. The contents of buffer[3], assigning the Peer Processor Timer state, override the contents of the Running or Not Running byte in buffer[1]:

buffer[3] = 1: ignore contents of buffer[1].
buffer[3] = 0: set Running or Not Running per buffer[1].

3. When the host application is set to Not Running, the adapter will ignore the following data configuration and transfer commands:

```
ncb_put_peer_cop()
ncb_xfer_glob_inp()
ncb_xfer_spec_inp()
ncb_xfer_spec_out().
```

# A.16 ncb_set_slave_login( )

### NetBIOS Command:

None

### Description

Sets or clears login status on a Program Slave path in the local adapter to the local host application. Once the path is established it can be used by a login command and subsequent programming commands from a remote application.

### Path Required

Requires the existence of a PS path.

### Syntax

```
int ncb_set_slave_login(*ncbp, login_status)
NCB *ncbp ;
unsigned char login_status ;
```

**Arguments**

| | |
|---|---|
| ncbp | A pointer to the NCB returned by the previous ncb_open(). |
| login_status | The desired login status: 1 sets login. 0 clears login. |

**Return Values**

| | |
|---|---|
| ERR_success | Good |
| ERR_busy | Error or adapter busy |
| ERR_bad_name_num | Bad NCB_NUM field |
| ERR_no_listen | Path is inactive |

# A.17 ncb_set_sw_interrupt( )

### NetBIOS Command

None

### Description

Sets the software interrupt vector used to access the device driver to the value passed to the function.

### Path Required

None

### Syntax

```
int ncb_set_sw_interrupt(swInterrupt)
int swInterrupt ;
```

**Argument**

| | |
|---|---|
| `swInterrupt` | The desired software interrupt vector value. Valid arguments are: 3...7, 10...FF. The vector value must match the value specified in the CONFIG.SYS file. |

**Return Values**

| | |
|---|---|
| `ERR_success` | Good |
| `ERR_invalid` | Did not affect the current interrupt vector setting |

# A.18 ncb_status( )

### NetBIOS Command

STATUS

### Description

Sets the host based device Run state and timeout parameters defined by a previous ncb_set_peer_params() function, and receives the current configuration. The user must allocate an NCB for the function that is separate from NCBs allocated by ncb_open().

The Run state and timeout value must first be assigned to the adapter by the ncb_set_peer_params() function. This ncb_status() function then makes the Run state and timeout value effective in the peer processor.

The function will receive a total of 110 bytes of status. The message buffer pointer in NCB_BUFFER must point to a valid buffer area of at least 110 bytes in length.

The first two bytes received into the buffer will be a diagnostic pattern 0xAA, 0x55 (alternating binary 0 and 1 in each bit position). This verifies that each bit position can be set and reset.

Bytes 2 - 109 will contain the received status information. This is the same status as is returned by Modbus function 8, subfunction 21 (Get/Clear Network Statistics). See the description of that function in Appendix F for a listing of status values.

### Path Required

Not applicable. The function does not require a path.

### Syntax

```
int ncb_status(*ncbp, adaptno)
NCB *ncbp ;
int adaptno ;
```

**Arguments**

| | |
|---|---|
| ncbp | A pointer to the user-allocated NCB. |
| adaptno | The host adapter device number—must be either 0 or 1. |

**Return Values**

| | |
|---|---|
| ERR_success | Good |
| ERR_invalid | Diagnostic failed |

# A.19 ncb_xfer_glob_inp( )

### NetBIOS Command

XFER_GLBINP

### Description

Transfers all newly received peer copped global input data from the network to the user's buffer.

**Buffer Data:** The buffer will contain data only from the nodes for which this peer cop application has actually been configured to receive data. For example if this processor is configured to receive one word of data from node 5 and two words of data from node 30, the first six bytes will consist only of data from those nodes with bytes 0, 1 from node 5, and bytes 2, 3, 4, 5 from node 30. The command does not zero-fill the buffer with data for unconfigured nodes.

**Buffer Flags:** The buffer will contain health flag bits starting at byte 1000, and data present flag bits starting at byte 1009. Byte 1000 is a flag bit indicating that the contents of the health table are valid or not valid. For each health flag bit, a logic 1 indicates that the remote node is active. For each data present flag bit, a logic 1 indicates that global input data is present from that node. Bits will be zero-filled for unconfigured nodes.

### Path Required

Not applicable. The function does not require a path.

### Syntax

```
int far ncb_xfer_glob_inp (NCB far * ncbp, int adaptno, char far*
buffer)
NCBP far* ncbp;
int adaptno;
char far *buffer;
```

**Argument**

| | |
|---|---|
| `ncbp` | A pointer to the NCB returned by the previous ncb_open(). |
| `adaptno` | The host device adapter number – must be 0 or 1 |
| `buffer` | A pointer to the user buffer (see Buffer Format) |

**Return Values**

| | |
|---|---|
| `ERR_success` | Normal completion |
| `ERR_invalid` | Data not transferred |

### Buffer Format

Command buffer:  Not applicable

Response buffer:
byte 0:  global input data from first configured node, first word lo-byte
byte 1:  global input data from first configured node, first word hi-byte

byte 2:  global input data from next configured node, last word lo-byte
byte 3:  global input data from next configured node, last word hi-byte
. . .
. . .  global input data from remaining configured nodes
. . .
byte 1000:  global input health table valid status (Note 1)
byte 1001:  global input health table, nodes 01-08
byte 1002:  global input health table, nodes 09-16
byte 1003:  global input health table, nodes 17-24
byte 1004:  global input health table, nodes 25-32
byte 1005:  global input health table, nodes 33-40
byte 1006:  global input health table, nodes 41-48
byte 1007:  global input health table, nodes 49-56
byte 1008:  global input health table, nodes 57-64
byte 1009:  global input present table, nodes 01-08
byte 1010:  global input present table, nodes 09-16
byte 1011:  global input present table, nodes 17-24
byte 1012:  global input present table, nodes 25-32
byte 1013:  global input present table, nodes 33-40
byte 1014:  global input present table, nodes 41-48
byte 1015:  global input present table, nodes 49-56
byte 1016:  global input present table, nodes 57-64

### Notes

1. Only the Most Significant Bit is implemented in this byte. If this bit is 1, then the returned global input health table is valid. If this bit is 0, then the table is not valid. The bit is set after the first complete token rotation cycle has been completed after an ncb_put_peer_cop() command has been executed. The bit is also set if this peer node is not in the normal token operation state, in which case all health bits are set to 0.

# A.20 ncb_xfer_spec_inp( )

### NetBIOS Command

XFER_SPCINP

### Description

Transfers all newly received peer copped specific input data from the network to the user's buffer.

**Buffer Data:** The buffer will contain data only from the nodes for which this peer cop application has actually been configured to receive data. For example if this processor is configured to receive one word of data from node 5 and two words of data from node 30, the first six bytes will consist only of data from those nodes with bytes 0, 1 from node 5, and bytes 2, 3, 4, 5 from node 30. The command does not zero-fill the buffer with data for unconfigured nodes.

**Buffer Flags:** The buffer will contain health flag bits starting at byte 1000, and data present flag bits starting at byte 1009. Byte 1000 is a flag bit indicating that the contents of the health table are valid or not valid. For each health flag bit, a logic 1 indicates that the remote node is active. For each data present flag bit, a logic 1 indicates that global input data is present from that node. Bits will be zero-filled for unconfigured nodes.

### Path Required

Not applicable. The function does not require a path.

### Syntax

```
int far ncb_xfer_spec_inp (NCB far * ncbp, int adaptno, char far*
buffer)
NCBP far* ncbp;
int adaptno;
char far *buffer;
```

| Argument | |
|---|---|
| ncbp | A pointer to the NCB returned by the previous ncb_open(). |
| adaptno | The host device adapter number – must be 0 or 1 |
| buffer | A pointer to the user buffer (see Buffer Format) |

| Return Values | |
|---|---|
| ERR_success | Normal completion |
| ERR_invalid | Data not transferred |

### Buffer Format

Command buffer:  Not applicable.

Response buffer:
byte 0:  specific input data from first configured node, first word lo-byte
byte 1:  specific input data from first configured node, first word hi-byte

byte 2:  specific input data from next configured node, last word lo-byte
byte 3:  specific input data from next configured node, last word hi-byte
. . .
. . .  specific input data from remaining configured nodes
. . .
byte 1000:  specific input health table valid status (Note 1)
byte 1001:  specific input health table, nodes 01-08
byte 1002:  specific input health table, nodes 09-16
byte 1003:  specific input health table, nodes 17-24
byte 1004:  specific input health table, nodes 25-32
byte 1005:  specific input health table, nodes 33-40
byte 1006:  specific input health table, nodes 41-48
byte 1007:  specific input health table, nodes 49-56
byte 1008:  specific input health table, nodes 57-64
byte 1009:  specific input present table, nodes 01-08
byte 1010:  specific input present table, nodes 09-16
byte 1011:  specific input present table, nodes 17-24
byte 1012:  specific input present table, nodes 25-32
byte 1013:  specific input present table, nodes 33-40
byte 1014:  specific input present table, nodes 41-48
byte 1015:  specific input present table, nodes 49-56
byte 1016:  specific input present table, nodes 57-64

### Notes

1.  Only the Most Significant Bit is implemented in this byte.  If this bit is 1, then the returned specific input health table is valid.  If this bit is 0, then the table is not valid.  The bit is set after the first complete token rotation cycle has been completed after an ncb_put_peer_cop() command has been executed.  The bit is also set if this peer node is not in the normal token operation state, in which case all health bits are set to 0.

# A.21 ncb_xfer_spec_out( )

### NetBIOS Command

XFER_SPCOUT

### Description

Transfers all peer copped specific output data from the user's buffer to the network.

**Buffer Data:** Load the buffer with data only for the nodes for which this peer processor has actually been configured to send data. For example if this peer cop application is configured to send one word of data to node 10 and two words of data to node 20, you should load bytes 0, 1 with data for node 10, and bytes 2, 3, 4, 5 for node 20. Do not zero-fill the buffer with data for unconfigured nodes.

**Buffer Flags:** The buffer contains health flag bits starting at byte 1000. Byte 1000 is a flag bit indicating that the contents of the health table are valid or not valid. For each health flag bit, a logic 1 indicates that the remote node is accepting the specific output data from this node.

### Path Required

Not applicable. The function does not require a path.

### Syntax

```
int far ncb_xfer_spec_out (NCB far * ncbp, int adaptno, char far*
buffer)
NCBP far* ncbp;
int adaptno;
char far *buffer;
```

| Argument | |
|---|---|
| ncbp | A pointer to the NCB returned by the previous ncb_open(). |
| adaptno | The host device adapter number – must be 0 or 1 |
| buffer | A pointer to the user buffer (see Buffer Format) |

| Return Values | |
|---|---|
| ERR_success | Normal completion |
| ERR_invalid | Data not transferred |

### Buffer Format

Command buffer:
byte 0:  specific output data to first configured node, first word lo-byte
byte 1:  specific output data to first configured node, first word hi-byte

byte 2:  specific output data to next configured node, last word lo-byte
byte 3:  specific output data to next configured node, last word hi-byte
. . .
. . .  specific output data to remaining configured nodes

Response buffer:
byte 1000: specific output health table valid status (Note 1)
byte 1001: specific output health table, nodes 01-08
byte 1002: specific output health table, nodes 09-16
byte 1003: specific output health table, nodes 17-24
byte 1004: specific output health table, nodes 25-32
byte 1005: specific output health table, nodes 33-40
byte 1006: specific output health table, nodes 41-48
byte 1007: specific output health table, nodes 49-56
byte 1008: specific output health table, nodes 57-64

### Notes

1. Only the Most Significant Bit is implemented in this byte. If this bit is 1, then the returned specific output health table is valid. If this bit is 0, then the table is not valid. The bit is set after the first complete token rotation cycle has been completed after an ncb_put_peer_cop() command has been executed. The bit is also set if this peer node is not in the normal token operation state, in which case all health bits are set to 0.

# Appendix B.
# Modifying the 5C Interrupt

## B.1 Modifying the 5C Interrupt

The software interrupt number to be used in communicating to the device driver (DOS applications only) is contained in the variable sw_interrupt, which is declared in the NETLIB.C code module.

The interrupt number is not applicable when using OS/2.

### B.1.1 Interrupts in Your Application

To change the software interrupt in your application, first refer to the chapter in this guide for installing the DOS software, and modify the interrupt parameter in the DEVICE line in your CONFIG.SYS file. Then have your application change the contents of the sw_interrupt variable to contain the new software interrupt number.

The following code fragment illustrates how to do this:

```
/* main.c – arg1 is the software interrupt number
usage: main /s5C */

extern int sw_interrupt;

int main( int argc, char *argv[] )
{
/* read command line parameter – if not assigned,
assign default */

if(sscanf( argv[1], ”/S%x”, &sw_interrupt) != 1 )
    sw_interrupt = 0x5c ;
}
```

**Using Windows DLLs:** Note that the above code will not work with applications that use Windows DLLs. In these applications, use the ncb_set_sw_interrupt() function described below.

You can also modify the interrupt within your application by a call to the ncb_set_sw_interrupt() function, described in Appendix A. You can find an example of the use of this function in the sample program BDRESET.C.

### B.1.2 Interrupts in MBPSTAT

As supplied, your Network Diagnostic Utility program MBPSTAT accepts the /S command argument for the software interrupt. When calling MBPSTAT, pass it the argument you used in the DEVICE line in your CONFIG.SYS file.

### B.1.3 Interrupts in Your Sample Programs

As supplied, your host based device board reset programs SA85OFF and BDRESET accept the /S command argument for the software interrupt. When calling either of these programs, pass it the argument you used in the DEVICE line in your CONFIG.SYS file.

Your other sample programs are supplied as coding examples, rather than as standalone programs capable of being run directly on your network configuration. Before running any program, refer to the program descriptions in Appendix E, including the caution about modifying the interrupt and node addresses in the programs. You can use the code fragment on the previous page as an example for editing the source code to use a different interrupt.

# Appendix C.
# Include Files

- □ netbios.h
- □ netlib.h

# C.1 netbios.h

```
/*============================================================================
 *      MODICON, Inc., 1 High Street, North Andover, MA 01845, USA
 *      Copyright (c) $Date: 1994/05/02 14:51:32 $ – All rights reserved.
 *
 *      No part of this document may be reproduced in any form
 *      without the express written consent of MODICON, Inc.
 *
 *=fdoc=======================================================================
 *
 * File:        netbios.h
 *
 *
 * Notes:       Include file for NetBIOS calls and definitions
 *
 * Author:
 *
 * RCSID:
 *
 * RCS:
 * $Revision: 3.33 $
 * $State: R3_4B1 $
 * $Locker:  $
 *
 * Revision History Log:
 *
 * Revision 3.32  03/11/94
 * Baseline
 *
 *=fdoc=====================================================================*/
```

```
#if defined(__OS2__) && defined(__32BIT__)
    #pragma seg16( NCB )
    #define FARPTR   * _Seg16
    #define LINKAGE _Optlink
    #define OS216ENTRY _Far16 _Cdecl
#else
    #define FARPTR far *
    #define LINKAGE
    #define OS216ENTRY
#endif

/* Structure of Network Control Block (NCB) */
typedef struct
    {
    unsigned char NCB_COMMAND;       /* command */
    unsigned char NCB_RETCODE;       /* function return code */
    unsigned char NCB_LSN;           /* local session number */
    unsigned char NCB_NUM;           /* number of network name */
    char FARPTR NCB_BUFFER;          /* far pointer to message buffer */
    unsigned short NCB_LENGTH;       /* length of message buffer */
    unsigned char NCB_CALLNAME[16];  /* name of session user is talking to */
    unsigned char NCB_NAME[16];      /* user’s network name  */
    unsigned char NCB_RTO;           /* receive time–out in 500 ms. incrs. */
    unsigned char NCB_STO;           /* send time–out – 500 ms. increments */
    char FARPTR NCB_POST_ADDRESS;    /*offset of ”no–wait” interrupt call */
    unsigned char NCB_LANA_NUM;      /* adapter number (must be 0 or 1) */
    unsigned char NCB_CMD_CPLT;      /* command completion status */
    unsigned char NCB_RESERVE[14];   /* Reserved area for Token–Ring */
    } NCB;
```

```
/* NetBIOS error return codes – returned in NCB_RETCODE */
#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 */
```

```
/* 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 */
```

## C.2 netlib.h

```
/*============================================================================
 *      MODICON, Inc., 1 High Street, North Andover, MA 01845, USA
 *      Copyright (c) $Date: 1994/05/02 14:49:52 $ – All rights reserved.
 *
 *      No part of this document may be reproduced in any form
 *      without the express written consent of MODICON, Inc.
 *
 *=fdoc=======================================================================
 *
 * File:  This header file contains the NETBIOS Modbus PLUS library prototypes.
 *
 * Notes:
 *
 * Author:
 *
 * RCS:
 * $Revision: 1.1 $
 * $State: R3_4B1 $
 * $Locker:  $
 *
 * Revision History Log:
 *
 *=fdoc=====================================================================*/
```

```
int far  ncb_reset(int adaptno);
int far  ncb_sa85off(int adaptno);
int far  ncb_status(NCB  far *ncbp,int adaptno);
int far  ncb_send(NCB  far *ncbp, int length, char  far *buffer, unsigned char timeout);
int far  ncb_receive_wait(NCB  far *ncbp, char  far *buffer, unsigned char timeout);
NCB  far * far  ncb_open(char  far *name, int lan);
int far  ncb_receive(NCB  far *ncbp, char  far *buffer);
int far  ncb_close(NCB  far *ncbp);
int far  ncb_send_datagram(NCB  far *ncbp, int length, char  far *buffer, unsigned char
timeout, int adaptno);
int far  ncb_receive_datagram(NCB  far *ncbp, int node, char  far *buffer, unsigned char
timeout, int adaptno);
int far  ncb_cancel(NCB  far *ncbp);
int far  ncb_set_slave_login(NCB  far *ncbp, unsigned char login_status);
int far  ncb_set_sw_interrupt (int swInterrupt);
void  exec_netbios(NCB  far *ncbstruct);

#ifndef SW_INTERRUPT
#define SW_INTERRUPT        0x5c         /* actual SW interrupt used by NETBIOS */
#endif
```

# Appendix D.
# The MBPSTAT Utility

# D.1 The MBPSTAT Program

MBPSTAT is the Modbus Plus network diagnostic utility. You can use it to create a list of active nodes, monitor overall activity on a network, and record error statistics from a specific node.

## D.1.1 Starting the Program

### DOS

When running under DOS, the program uses the standard NetBIOS interrupt 5C as a default. You can specify a different interrupt in a command line argument /**S**. The argument you enter must be the same as you configured for the DOS driver in your CONFIG.SYS file.

Change to the directory containing MBPSTAT.EXE. To start the program using the default interrupt 5C, enter: **MBPSTAT**

To use a different interrupt, enter it on the command line as in the following example: **MBPSTAT /S5B** or **MBPSTAT /S5D**

### OS/2

Change to the directory containing MBPSTAT.EXE. To start the program, enter: **MBPSTAT**

### Board Number

When the program starts, it will prompt you for the board number for the host based device from which you want to run the network diagnostic. Enter the board number (0 or 1) you configured in your computer's CONFIG.SYS file during installation of the device driver.

## D.1.2 Selecting the Network to be Analyzed

The program will next prompt you for a routing path to the Modbus Plus network you want to analyze. For example, you can examine the local network, or a remote network that is accessed through a series of Bridge Plus devices. Note that you are selecting a target network only -- the address of any specific node on that network will be entered later as you run the program.

You can enter up to four network routing bytes. The last non-zero entry defines the target network. The first zero entry terminates the routing. Some routing examples are shown on the next page.

### D.1.3 Network Selection Examples

If you enter a routing path of all zeros, you are disabling routing to other networks. You are instructing the program to analyze the local Modbus Plus network (the one to which your local host based device is directly connected).

If you enter a routing path of 22 00 00 00, you are instructing the program to analyze a second network that is connected to the remote side of a Bridge Plus device, whose address is 22 on the local network.

If you enter 22 24 00 00, you are instructing the program to analyze a third network that is connected through two Bridge Plus devices -- through address 22 on the local network, and then through address 24 on the second network.

While running the program, you can change the routing path to select another network through an option on your program menu.

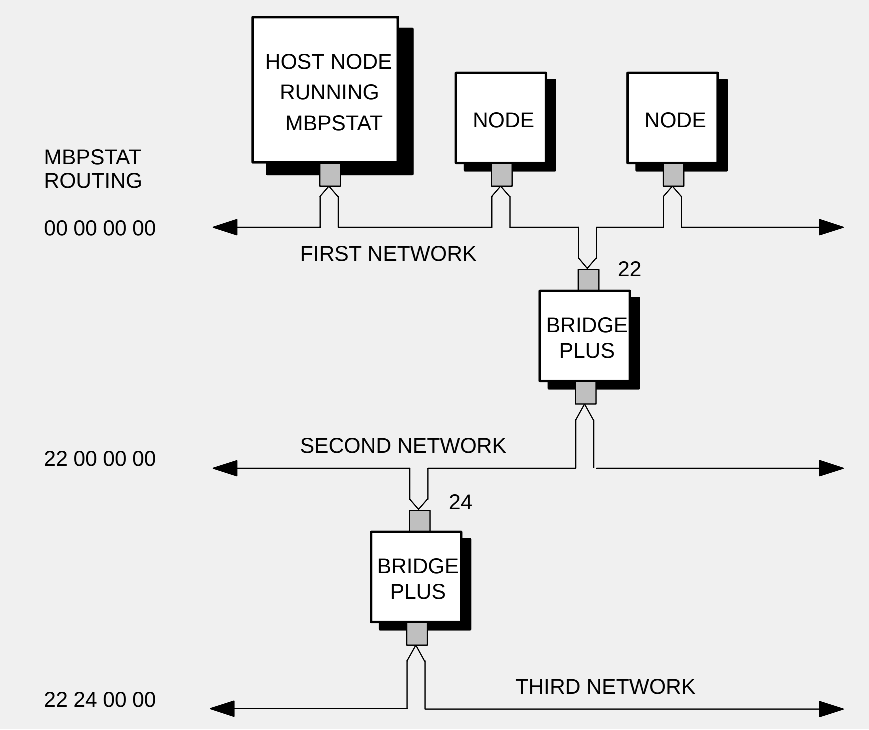
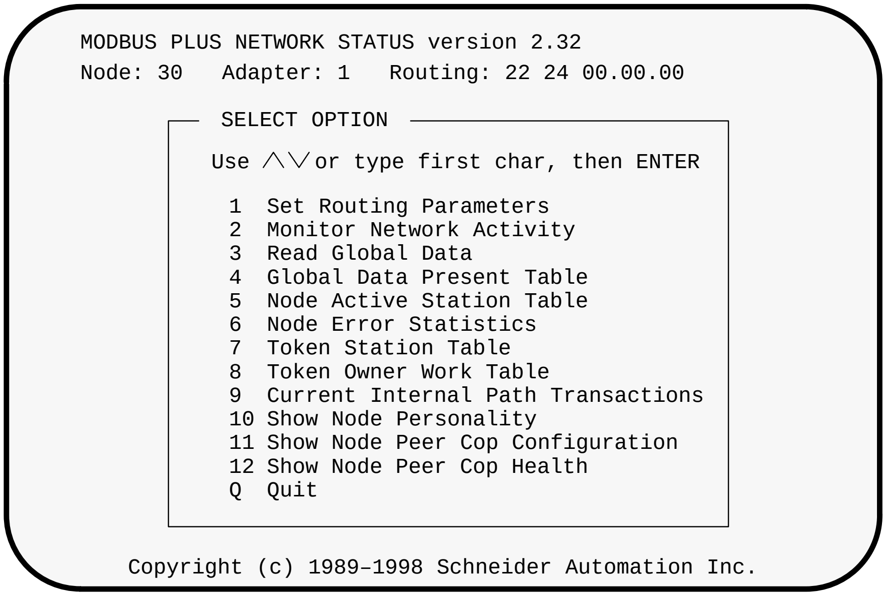


**Figure 32 Typical MBPSTAT Routing**

# D.2 The MBPSTAT Options Menu

After you have entered the board number and network routing path, your Options menu appears. The local node address and board number appear at the top of the menu. The network routing path being analyzed is also shown.



```
MODBUS PLUS NETWORK STATUS version 2.32
Node: 30   Adapter: 1   Routing: 22 24 00.00.00

        SELECT OPTION

          Use ∧∨ or type first char, then ENTER

           1  Set Routing Parameters
           2  Monitor Network Activity
           3  Read Global Data
           4  Global Data Present Table
           5  Node Active Station Table
           6  Node Error Statistics
           7  Token Station Table
           8  Token Owner Work Table
           9  Current Internal Path Transactions
           10 Show Node Personality
           11 Show Node Peer Cop Configuration
           12 Show Node Peer Cop Health
           Q  Quit

    Copyright (c) 1989–1998 Schneider Automation Inc.
```

**Figure 33 MBPSTAT Options Menu**

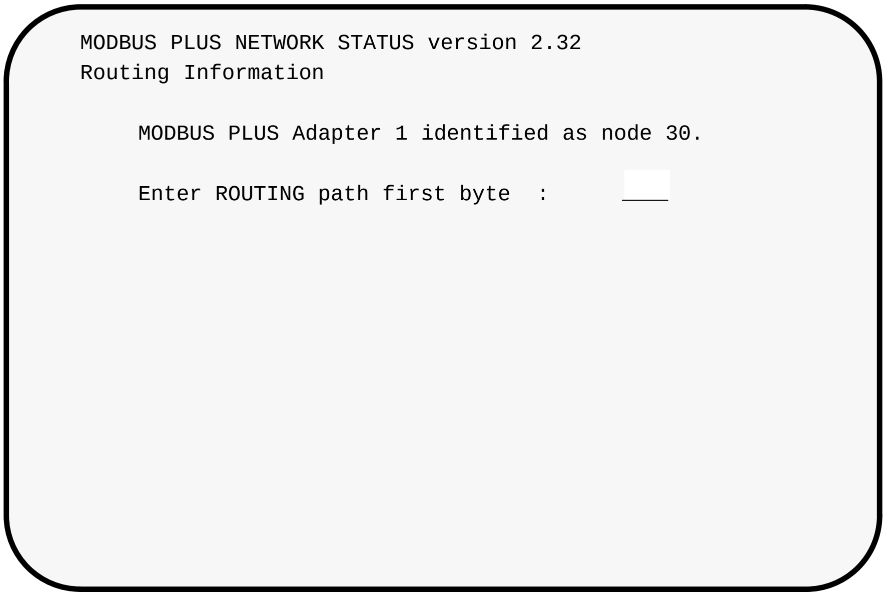
To select an option, use the cursor keys to move to the option, then press ENTER. You can also type in the option (1-10 or Q) and press ENTER. To terminate any test and return to your options menu, press ESCAPE.

### Entering Node Addresses

Some options analyze an overall network, and do not require selection of a node. Other options analyze the activity of a single node. For those options, the program will prompt you for a node address. When prompted, enter the address of the node you wish to examine. Note that any node being analyzed will always be located on the network you previously specified as the network routing path.

# D.3 MBPSTAT Options

## D.3.1 Option 1: Set Routing Parameters

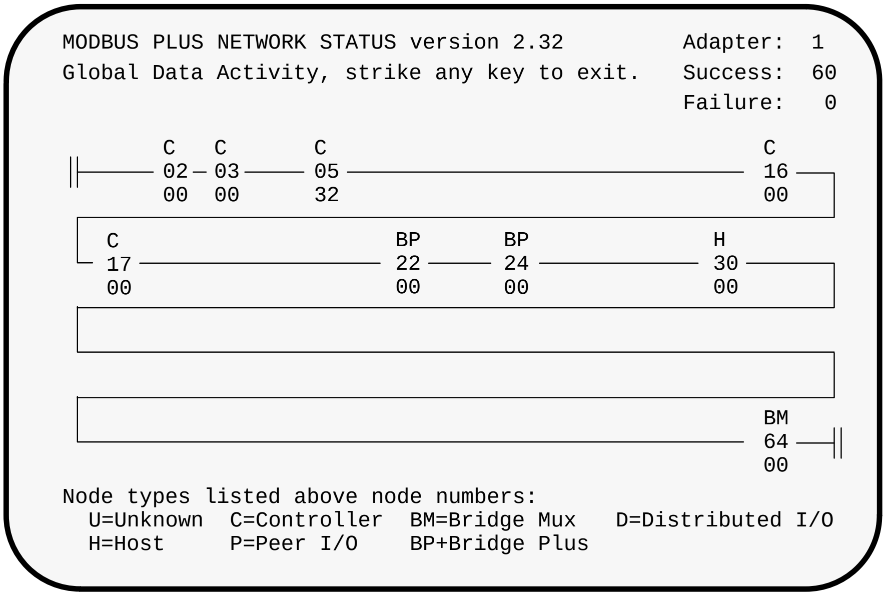


```
MODBUS PLUS NETWORK STATUS version 2.32
Routing Information

    MODBUS PLUS Adapter 1 identified as node 30.

    Enter ROUTING path first byte  :      ____
```

Figure 34 Set Routing Parameters

This option lets you specify a new routing path without having to restart MBPSTAT. The program will prompt you for each byte of routing.

### D.3.2 Option 2: Monitor Network Activity



```
MODBUS PLUS NETWORK STATUS version 2.32         Adapter:  1
Global Data Activity, strike any key to exit.   Success:  60
                                                Failure:   0

        C   C       C                                  C
        02  03      05                                 16
        00  00      32                                 00

    C                      BP       BP               H
    17                     22       24               30
    00                     00       00               00




                                                       BM
                                                       64
                                                       00
Node types listed above node numbers:
  U=Unknown  C=Controller  BM=Bridge Mux   D=Distributed I/O
  H=Host     P=Peer I/O    BP+Bridge Plus
```

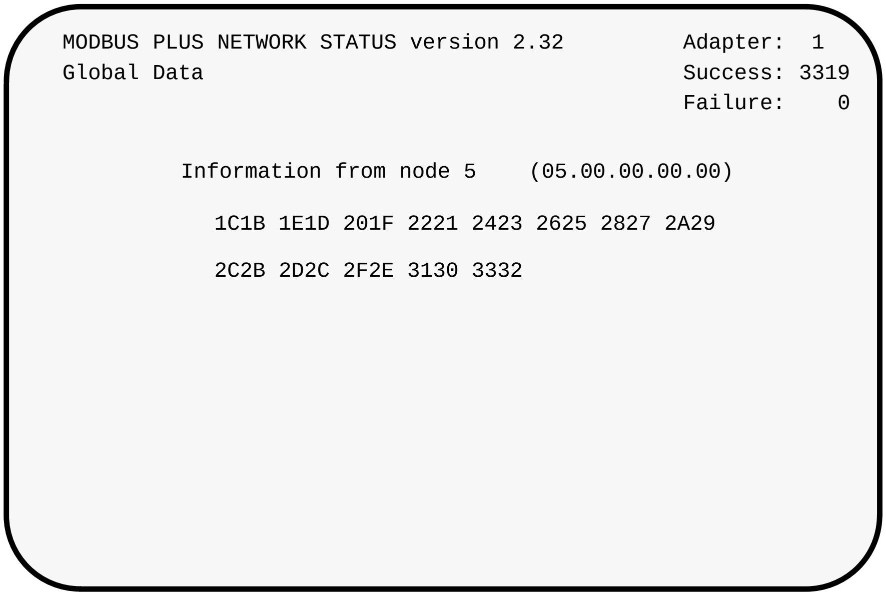
Figure 35 Monitor Network Activity

This option lists the nodes that exist on the network specified by the routing path. The program attempts to communicate with every node from address 1 to 64, and checks whether a response has been received from each node.

The program also attempts to read global data from each active node and displays the number of words of global data read. The global data word count is displayed beneath each node's address. Zeros will be displayed if no global data was read. If you are accessing a remote network through a Bridge Plus, the program will inform you that global data is not accessible through the bridge.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

## D.3.3 Option 3: Read Global Data



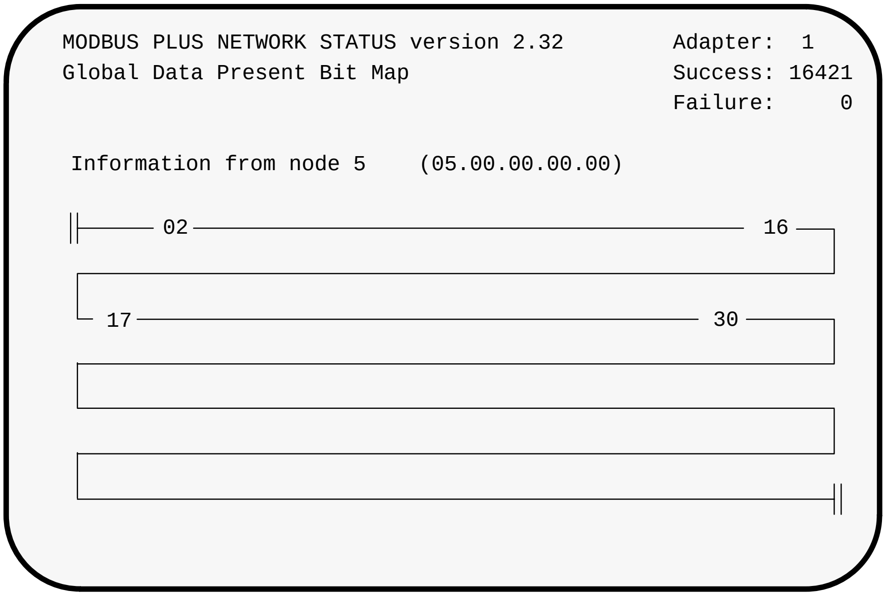
```
MODBUS PLUS NETWORK STATUS version 2.32         Adapter:  1
Global Data                                     Success: 3319
                                                Failure:    0

         Information from node 5    (05.00.00.00.00)

            1C1B 1E1D 201F 2221 2423 2625 2827 2A29

            2C2B 2D2C 2F2E 3130 3332
```

Figure 36 Read Global Data

This option continuously reads and displays global data from the selected node. If global data is present but is zero words in length, a 'null data' message is displayed.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

### D.3.4 Option 4: Global Data Present Table



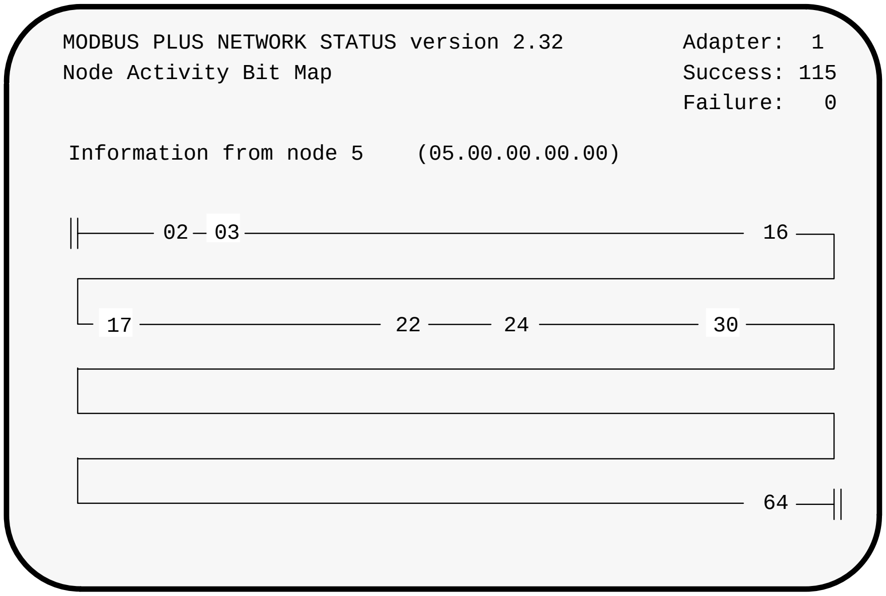
```
MODBUS PLUS NETWORK STATUS version 2.32        Adapter:  1
Global Data Present Bit Map                    Success: 16421
                                               Failure:     0

 Information from node 5    (05.00.00.00.00)

      02                                      16

   17                                      30
```

**Figure 37 Global Data Present Table**

This option displays a map table of the nodes that have global data present, as seen by the specified node. Node addresses appearing in the table are those from which the specified node has received global data since the test was started. The nodes in the table are not necessarily the only ones active on the network.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

## D.3.5 Option 5: Node Active Station Table



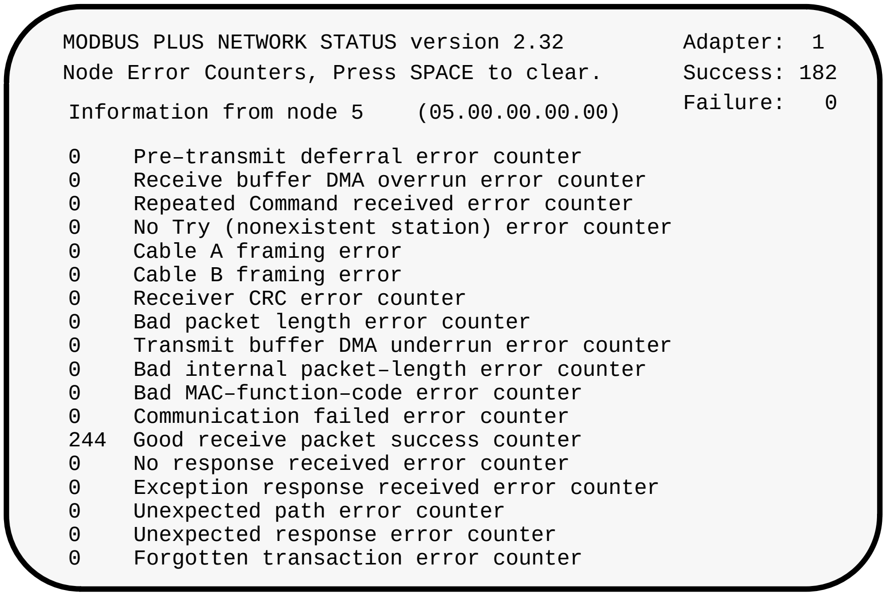
```
MODBUS PLUS NETWORK STATUS version 2.32         Adapter:  1
Node Activity Bit Map                           Success: 115
                                                Failure:   0

 Information from node 5    (05.00.00.00.00)


      02  03                                         16

   17                     22     24            30



                                                     64
```

Figure 38 Node Active Station Table

This option continuously queries the specified node for its list of active nodes and displays that list. Node addresses shown in normal video are those seen by the node as remaining active during the test. Node addresses in reverse video are those which were active at least once during the test, but which have subsequently gone inactive.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

### D.3.6 Option 6: Node Error Statistics



```
MODBUS PLUS NETWORK STATUS version 2.32         Adapter:  1
Node Error Counters, Press SPACE to clear.      Success: 182
                                                Failure:   0
 Information from node 5    (05.00.00.00.00)

 0    Pre–transmit deferral error counter
 0    Receive buffer DMA overrun error counter
 0    Repeated Command received error counter
 0    No Try (nonexistent station) error counter
 0    Cable A framing error
 0    Cable B framing error
 0    Receiver CRC error counter
 0    Bad packet length error counter
 0    Transmit buffer DMA underrun error counter
 0    Bad internal packet–length error counter
 0    Bad MAC–function–code error counter
 0    Communication failed error counter
 244  Good receive packet success counter
 0    No response received error counter
 0    Exception response received error counter
 0    Unexpected path error counter
 0    Unexpected response error counter
 0    Forgotten transaction error counter
```

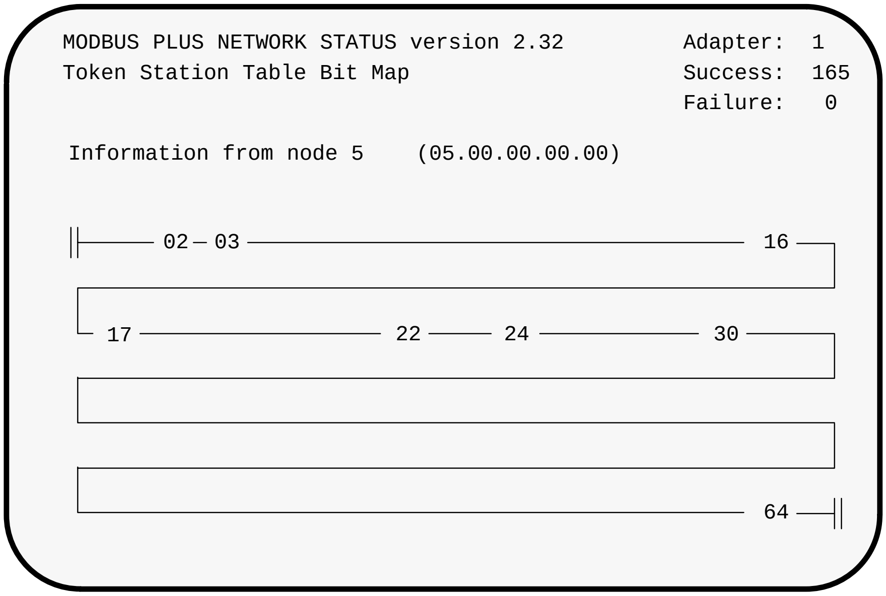
**Figure 39 Node Error Statistics**

This option continuously queries the specified node for a list of its error statistics and displays the list. As the test runs, the counts in the display can be reset to zeros by pressing the SPACE bar.

If you are running the utility on a single-cable network, one of the 'framing error' counters will be incrementing continuously due to the open connection on the monitoring device's other cable connector. For example, if your single cable is connected to the Cable A connector, the 'Cable B framing error' counter will increment continuously. In such cases you can disregard the counter.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

## D.3.7 Option 7: Token Station Table



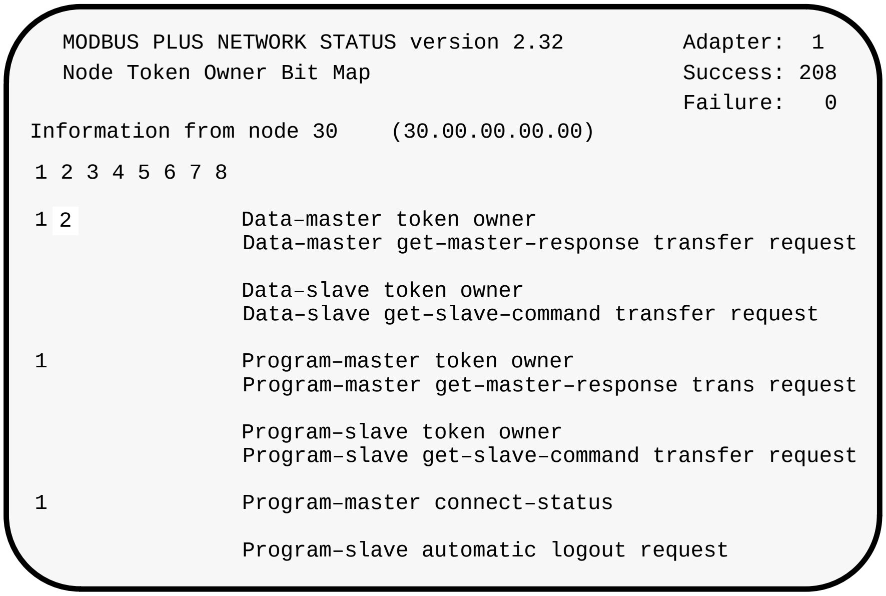
```
MODBUS PLUS NETWORK STATUS version 2.32          Adapter:  1
Token Station Table Bit Map                      Success:  165
                                                 Failure:   0

 Information from node 5    (05.00.00.00.00)

02 03 16
17 22 24 30
64
```

**Figure 40 Token Station Table**

This option displays a map table of the nodes that are passing the network token, as seen by the specified node.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

### D.3.8 Option 8: Token Owner Work Table



```
  MODBUS PLUS NETWORK STATUS version 2.32        Adapter:  1
  Node Token Owner Bit Map                       Success: 208
                                                 Failure:   0
Information from node 30    (30.00.00.00.00)

1 2 3 4 5 6 7 8

1 2            Data–master token owner
               Data–master get–master–response transfer request

               Data–slave token owner
               Data–slave get–slave–command transfer request

1              Program–master token owner
               Program–master get–master–response trans request

               Program–slave token owner
               Program–slave get–slave–command transfer request

1              Program–master connect–status

               Program–slave automatic logout request
```

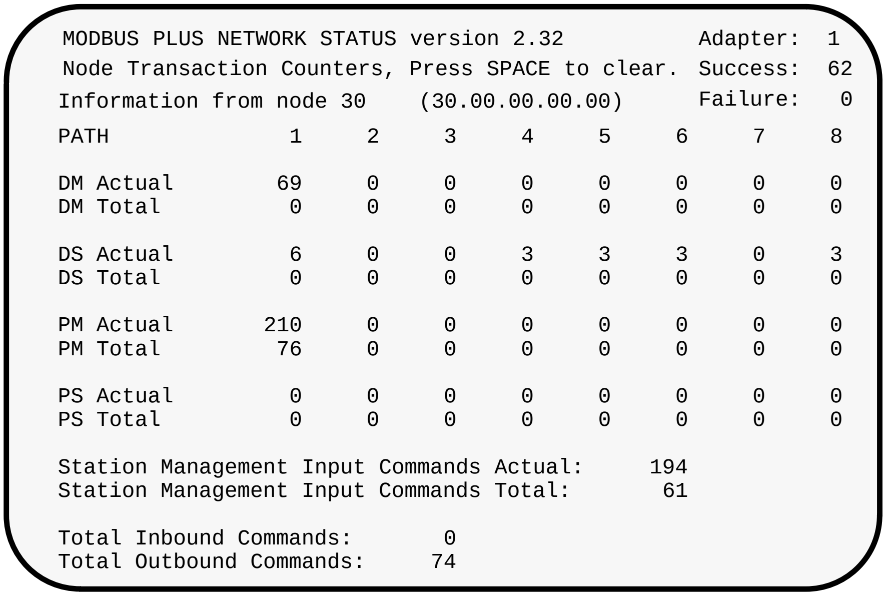
**Figure 41 Token Owner Work Table**

This option continuously queries the specified node for its list of internal paths that are active. The paths listed are the node’s internal data and program paths.

As a path goes active its number is listed in the columns to the left. Paths shown in normal video are those remaining active during the test. Paths in reverse video are those which were active at least once during the test, but which have subsequently gone inactive.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

## D.3.9 Option 9: Current Internal Path Transactions



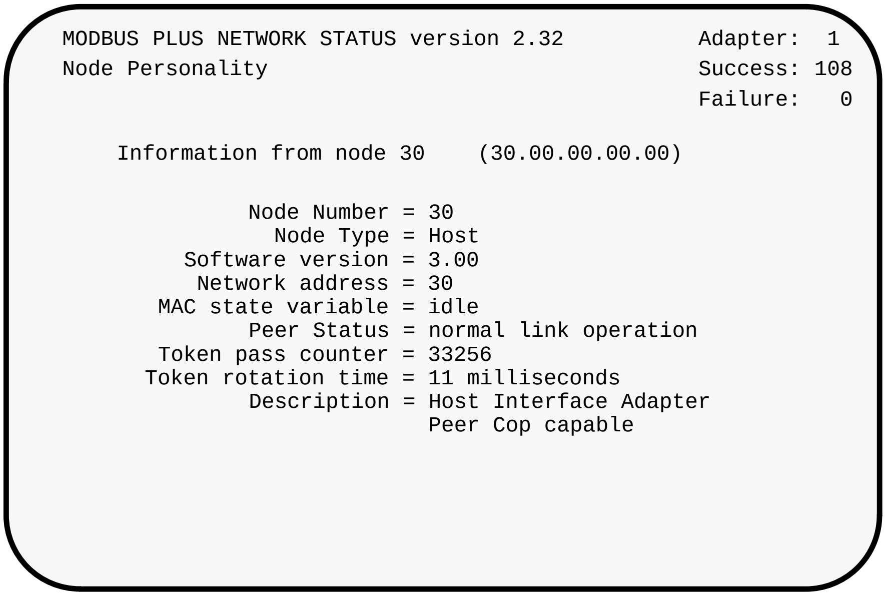
```
 MODBUS PLUS NETWORK STATUS version 2.32          Adapter:  1
 Node Transaction Counters, Press SPACE to clear.  Success:  62
Information from node 30    (30.00.00.00.00)      Failure:   0
PATH              1     2     3     4     5     6     7     8

DM Actual        69     0     0     0     0     0     0     0
DM Total          0     0     0     0     0     0     0     0

DS Actual         6     0     0     3     3     3     0     3
DS Total          0     0     0     0     0     0     0     0

PM Actual       210     0     0     0     0     0     0     0
PM Total         76     0     0     0     0     0     0     0

PS Actual         0     0     0     0     0     0     0     0
PS Total          0     0     0     0     0     0     0     0

Station Management Input Commands Actual:     194
Station Management Input Commands Total:       61

Total Inbound Commands:       0
Total Outbound Commands:     74
```

**Figure 42 Current Internal Path Transactions**

This option displays the count of transactions processed by each of the specified node's internal paths. As the test runs, the counts in the display can be reset to zeros by pressing the SPACE bar.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

## D.3.10 Option 10: Node Personality



```
MODBUS PLUS NETWORK STATUS version 2.32          Adapter:  1
Node Personality                                 Success: 108
                                                 Failure:   0

    Information from node 30    (30.00.00.00.00)

              Node Number = 30
                Node Type = Host
         Software version = 3.00
          Network address = 30
       MAC state variable = idle
              Peer Status = normal link operation
       Token pass counter = 33256
      Token rotation time = 11 milliseconds
              Description = Host Interface Adapter
                            Peer Cop capable
```

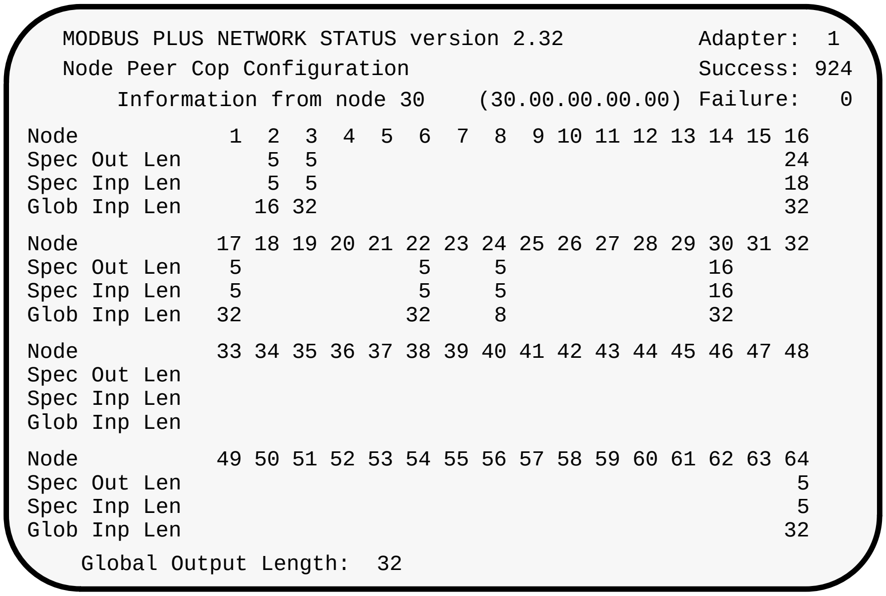
**Figure 43 Node Personality**

This option displays information about the specified node, such as its node type, software version, and ability to handle Peer Cop data. It also shows information about the node's current activity on the network.

Note that the Peer Cop capability applies to the AM-SA85-002 only.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

### D.3.11 Option 11: Node Peer Cop Configuration



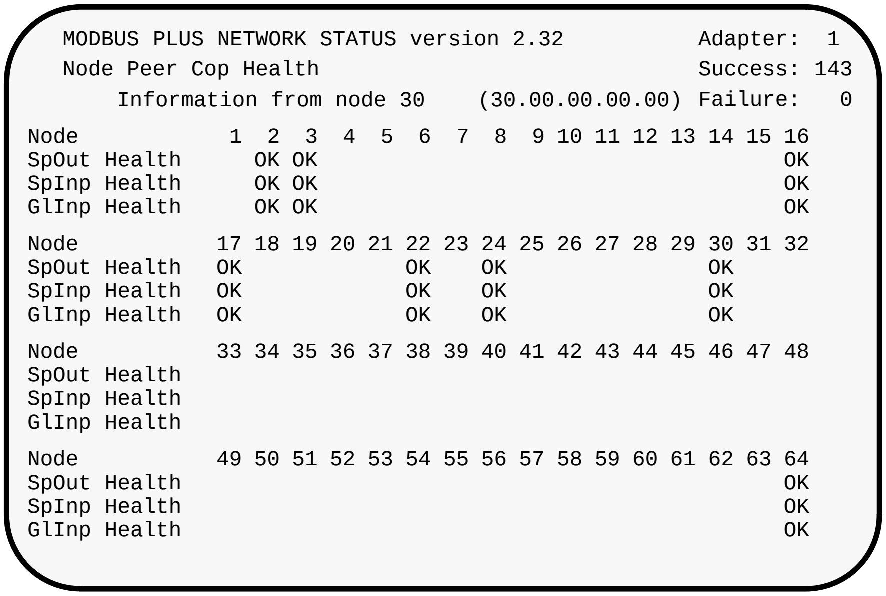
```
   MODBUS PLUS NETWORK STATUS version 2.32           Adapter:  1
   Node Peer Cop Configuration                       Success: 924
       Information from node 30     (30.00.00.00.00) Failure:   0

Node               1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
Spec Out Len          5  5                                     24
Spec Inp Len          5  5                                     18
Glob Inp Len         16 32                                     32

Node              17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Spec Out Len       5              5     5                 16
Spec Inp Len       5              5     5                 16
Glob Inp Len      32             32     8                 32

Node              33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
Spec Out Len
Spec Inp Len
Glob Inp Len

Node              49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
Spec Out Len                                                    5
Spec Inp Len                                                    5
Glob Inp Len                                                   32

    Global Output Length:  32
```

**Figure 44 Node Peer Cop Configuration**

This option displays information about the Peer Cop Configuration of each active node, as seen by the selected node.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

## D.3.12 Option 12: Node Peer Cop Health



```
   MODBUS PLUS NETWORK STATUS version 2.32           Adapter:   1
   Node Peer Cop Health                              Success: 143
       Information from node 30      (30.00.00.00.00) Failure:   0

Node             1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
SpOut Health        OK OK                                     OK
SpInp Health        OK OK                                     OK
GlInp Health        OK OK                                     OK

Node           17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
SpOut Health   OK             OK    OK                OK
SpInp Health   OK             OK    OK                OK
GlInp Health   OK             OK    OK                OK

Node           33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
SpOut Health
SpInp Health
GlInp Health

Node           49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
SpOut Health                                                OK
SpInp Health                                                OK
GlInp Health                                                OK
```

**Figure 45 Node Peer Cop Health**

This option displays information about the Peer Cop Health of each active node, as seen by the selected node.

The option runs continuously and displays a pass count until you terminate it by pressing ESCAPE.

## D.3.13 Option Q: Quit

This option exits MBPSTAT and returns you to your operating system prompt.

# Appendix E.
# Sample Programs

# E.1 Using the Sample Programs

Sample programs are provided on the disks supplied with your 416 NHM 212 00 Modbus Plus Adapter. They can be helpful for showing data transfers between Modbus Plus nodes in applications. The programs are TEST4.EXE, TEST4B.EXE, TEST4C.EXE, TESTSLAV, GLOBTEST, READNODE, MBPCHECK, and READNET.

In addition to the executable files, the complete C language source code files are provided. You can examine these as examples for coding.

⚠ **Caution: These programs are supplied as examples of coding. They are not supplied as standalone programs to be run on your network. Before running any program, you should examine its source code listing, modify any software interrupt and/or network node addresses in the programs to agree with your network configuration, and recompile the programs. If you run any program without modifying it for your configuration, unpredictable operation of your network may result.**

## E.1.1 Minimum Configuration Requirements

**MBPCHECK and READNET**
To run these programs, you will need one or two host based devices configured in the host computer and available on the network.

**GLOBTEST, TESTSLAV and TEST4C**
To run these programs, you will need two host based devices configured in separate host computers and available on the network.

**TEST4 and READNODE**
To run these programs, you will need one host based device configured in its host and available on the network, and one or more remote PLCs available on the network.

**TEST4B**
To run this program, you will need two host based devices configured in separate hosts and available on the network, and two or more remote PLCs available on the network.

## E.1.2 Interactive Programs

The GLOBTEST program generates global data at the local host node. Its purpose is to provide global data for testing at a second host node that is running MBPSTAT, the network diagnostic utility (see Appendix C).

The TEST4CandTESTSLAV programs are to be run simultaneously at two host nodes. These programs work together to cause data transfers between the two nodes across the network.

### E.1.3 Node Addresses

All of the programs except GLOBTEST take node addresses as command line arguments. GLOBTEST does not use an address.

The nodes you intend to use in these programs be present on the network and must be used by your application while the programs are running.

### E.1.4 Interrupt 5C (DOS)

Your sample programs are coded using the standard NetBIOS interrupt 5C. If you are using a different interrupt, you can use the code fragment in Appendix B to modify your programs, and recompile them. The device board reset programs, SA85OFF and BDRESET, accept command arguments for a different interrupt and can be run without modification.

### E.1.5 Starting a Program

To start one of the programs, change to the directory containing the executable files and get the operating system prompt for that directory. Enter the program's name together with any required command line arguments. Arguments are listed in the program descriptions on the following pages. If you do not enter the proper arguments, the program will prompt you for the correct entries.

# E.2 Program Descriptions

## E.2.1 MBPCHECK

**Starting Command**
MBPCHECK

**Function**
This program identifies the presence of one or two host devices in the local host. It reports the device number (0 or 1) and network node address (1 ... 64).

A typical message is:

```
MODBUS PLUS Adapter 0 identified as node 5.
```

The program reports this information and terminates.

## E.2.2 READNET

**Starting Command**
READNET  or READNET *routing_path*

**Function**
This program identifies the presence of all active nodes on the network. If reports each device's network node address and resident software version.

The command line argument *routing_path* is the network routing path to the network to be checked. Enter five bytes to specify the path to the network, with the bytes separated by periods ( . ).

For a complete description of usage, enter: **READNET ?**

If no command line argument is used, the host device's local link is checked. If a routing path is specified in the command line, the specified link is checked.

Typical messages are:

```
Station 5 is a Host, Software version 2.01.
Station 9 is a Controller, Software version 2.02.
```

The program reports this information and terminates.

## E.2.3 TEST4

**Starting Command**
TEST4 *slave_node*

**Function**
This program reads 125 holding registers from a PLC at the slave node specified in the command line. The registers to be read are 40001 through 40125. The program runs continuously until terminated by either <Control> c or <Control> <Break>.

For each successful read request of the 125 registers, the program displays received data.

## E.2.4 TEST4B

**Starting Command**
TEST4B *first_slave_node second_slave_node* [/d]

**Function**
If the /d (dual adapter flag) parameter is not specified, the default test mode is for one local adapter to communicate with two networked PLCs. It reads 125 holding registers from each of the specified slave nodes.

If the /d (dual adapter flag) parameter is specified, the program assumes the presence of two adapters configured as boards 0 and 1. It causes board 0 to read 125 holding registers from a programmable controller at the first slave node that is specified in the command line, and 125 registers from the controller at the second slave node specified in the command line.

The registers to be read are 40001 ... 40125 in the two PLCs. The program runs continuously until terminated by either (Control> c or <Control> <Break>.

For each successful request to read at the first node, the program displays ‘ + ’. For each successful read of 125 registers at board 0, the program displays ‘ - ’. For each successful request to read at the second node, the program displays ‘ < ’. For each successful read of 125 registers at board 1, the program displays ‘ > ’. A continuous display of successful passes should appear as:

```
+<–>+<–>+<–>
```

## E.2.5 TEST4C and TESTSLAV

**Starting Command**
TEST4C *slave_address*
TESTSLAV

**Function**
These programs are intended to be run simultaneously between two host devices. TEST4C reads 125 holding registers across the network from the device that is running TESTSLAV. The latter device is a Modbus slave, waiting for the incoming requests for data. The programs run continuously until terminated by <Control> c or <Control> <Break>.

For each successful request to read 125 registers, TEST4C displays ‘ + ’. For each successful read of 125 registers, TEST4C displays ‘ - ’. A continuous display of successful passes should appear as: `+–+–+–`

For each request it services, TESTSLAV displays ‘ + ’. A continuous display of successful passes should appear as: `++++++`

As it is unlikely that both programs will start at the same time, error messages could appear from one program while you are in the process of starting the other.

If TEST4C accesses an active node that is not running TESTSLAV, it will display `Receive error = 18,` indicating a routing failure.

If TESTSLAV starts while TEST4C is not running, it will display `Slave Status = 0x204` at approximately 15 s intervals, indicating that no request was received during that interval.

## E.2.6 GLOBTEST

**Starting Command**
GLOBTEST

**Function**
This program continuously generates global data from host device board 0. Its main purpose is to create global data activity that can be read and displayed by the ‘Read Global Data’ test in the Network Diagnostic Utility MBPSTAT.EXE. The program runs continuously until terminated by either <Control> c or <Control> <Break>.

## E.2.7 READNODE

**Starting Command**
READNODE *routing_path reference length count*

**Function**
This program reads and displays discrete inputs, coils, input registers, and holding registers from a specified PLC node. The program runs a specified number of passes and then terminates automatically, or else runs continuously until terminated by either <Control> c or <Control> <Break>.

The command line arguments are:

□ *routing_path*
The network routing path to the node to be read. Enter one to five bytes to specify the path to the destination node, with the bytes separated by periods ( . ).

For example, the single entry 10 specifies node 10 on the local network. The entry 24.10 specifies node 10 on a remote network that is connected to the local network by a Bridge Plus device at address 24. The entry 24.22.10 specifies node 10 on a remote network that is connected to the local network by two Bridge Plus devices. The first Bridge Plus is at address 24 on the local network. The second Bridge Plus is at address 22 on a second network.

□ *reference*
A base reference of 0*x*, 1*x*, 3*x*, or 4*x*.

□ *length*
The number of registers to be read starting at the base reference.

□ *count*
The number of times to read from the node. The program will terminate when the count value is reached A count value of 0 causes the program to read continuously until control-c or control-break is entered.

For example: `READNODE  24.5  40001  10  1` addresses node 5 on a remote network through a Bridge Plus device at node 24 on the local network. The program will read 10 registers starting at 40001, and will make one pass.

## E.3 Board Reset Programs

As long as the Modbus Plus adapter is enabled, its device driver continually polls it for new activity. If your application no longer needs the device, you may want to disable it to avoid timing conflicts with a new program.

Two programs are provided for disabling and restarting the Modbus Plus adapter. You can execute them at the operating system prompt, or call them from your application. The programs are: SA85OFF.EXE and BDRESET.EXE.

You can prevent the use of these programs by specifying the /B parameter in the MBPHOST command line.

### E.3.1 SA85OFF.EXE

SA85OFF.EXE turns off the device driver's polling of the Modbus Plus adapter. You would typically use this program when your application no longer requires the adapter and you want to use your computer product for other purposes. You can later call BDRESET.EXE to restore the adapter's operation.

The program's main purpose is to allow you to run terminal emulation programs on your computer without interference from the device driver polling. Otherwise, the driver will continue polling the adapter according to the internal timer, in which case the serial port might not be handled properly. For example, if you are using a Modicon P190 Programming Panel emulation program you might get 'UART Error' messages. Execute SA85OFF to turn off the driver before running your emulator.

Two arguments are used:

□ /S specifies an interrupt

□ ,/N specifies board 0 or 1

For example: `SA85OFF  /S5B /N1` uses interrupt 5B to disable board one. If an argument is not specified, the program's defaults are interrupt 5C and board zero.

### E.3.2 BDRESET.EXE

BDRESET.EXE initializes the host based device board, causes it to run its internal diagnostics, and places it into a known initial state. Upon successful initialization the board attempts to join the network. This program performs the same level of initialization as in power-up of the computer.

Two arguments are used:

- /S specifies an interrupt
- /N specifies board 0 or 1

For example: `BDRESET  /S5B /N1` uses interrupt 5B to initialize board one.

# Appendix F.
# Using Modbus Commands

# F.1 Modbus Protocol for Modbus Plus

This appendix describes Modbus commands as they are used in Modbus Plus. This information is intended for programmers who wish to write applications that communicate between the host based devices and Modicon PLCs. For complete details and examples of Modbus commands, see the *Modbus Protocol Reference Guide*.

Modicon PLCs use the Modbus protocol to access data and statistics. Modbus Plus uses a set of Modbus commands to perform data acquisition tasks between your host based device and Modicon 984 controllers across the network. The Modbus protocol determines how:

□ The message sender and receiver identify each other

□ The system maintains network-wide order as messages are exchanged

□ Errors on the network are detected.

## F.1.1 The Modbus Master–Slave Relationship

In the original Modbus protocol, only one master device can exist on a single network, originating contacts with multiple slave devices. The Modbus Plus peer-to-peer protocol allows multiple masters to contact multiple slave devices. Concurrent transactions are handled using multiple paths of various types in the devices.

The Modbus protocol uses a query-response cycle between master and slave devices. A complete message transaction consists of a query originated by a Modbus master device and a response sent by the slave back to the master.

## F.1.2 Creating Modbus Queries and Responses

A Modbus query message originated by a host application can be constructed in a data buffer. The message should consist of the Modbus function code and data elements in the formats described in this appendix. To send the message, the buffer can be accessed by the NetBIOS functions (see Appendix B). Similarly, a slave application in the host can receive the query into a buffer using NetBIOS functions. It can construct the response message in a buffer and send it.

The processes of obtaining a network path, designating a Modbus Plus destination node, and imbedding the Modbus message into a Modbus Plus message packet are handled by the NetBIOS functions and network hardware.

## F.2 The Modbus Transaction

The format for Modbus commands or responses is an eight-bit function code followed by a block of eight-bit bytes (maximum length: 252 bytes).

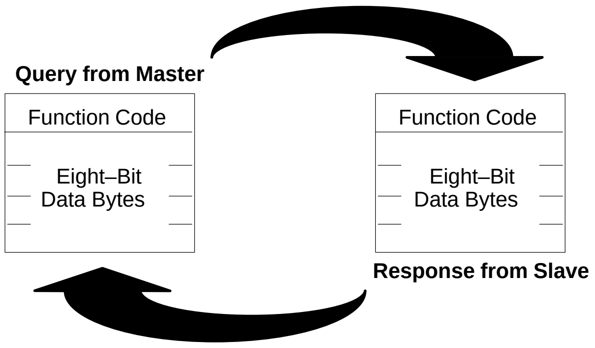


**Figure 46 Modbus Query and Response Cycle**

### The Query

The function code in the query tells the addressed slave device what kind of action to perform. The data bytes contain any additional information that the slave will need to perform the function. For example, if you issue function code 03, querying the slave to read holding registers and respond with their contents, the data field must contain information telling the slave which register to start at and how many registers to read.

### The Response

If the slave makes a normal response, the function code in the response is an echo of the function code in the query. The data bytes contain the data collected by the slave, such as register values or status. If an error occurs, the function code is modified to indicate that the response is an error response, and the data bytes contain a code that describes the error.

The buffer contents of the message as it is used for Modbus Plus are nearly the same as in a standard Modbus message, with two exceptions:

□ A Modbus message contains the slave device address. This is stripped from the Modbus message contents before transmission over Modbus Plus. It appears instead in the Modbus Plus MAC level destination field.

□ The Modbus CRC/LRC error check field is stripped from the message contents. Error checking is performed on the entire Modbus Plus message in the HDLC level CRC-16 field.

# F.3 Modbus Command Summary

| Code (Hex) | Meaning | User Action |
|---|---|---|
| 01 | *Read Coil Status* | Obtain the current ON/OFF status of a group of logic coils |
| 02 | *Read Input Status* | Obtain the current ON/OFF status of a group of discrete Inputs |
| 03 | *Read 4x Registers* | Obtain the current binary value in one or more holding registers |
| 04 | *Read 3x Registers* | Obtain the current binary value in one or more input registers |
| 05 | *Force Single Coil* | Force a logic coil ON or OFF |
| 06 | *Preset Single Register* | Place a specific binary value in a holding register |
| 07 | *Read Exception Status* | Obtain the current ON/OFF status of internal coils 00001 ... 00008 in a PLC; user logic determines the status of the coils; short message length allows rapid reading |
| 08 (sub 15 hex) | *Get/Clear Network Statistics* | Obtain current network statistics from the host; specify subfunction 21 (15 hex) |
| 09 ... 0E | Reserved | N/A |
| 0F | *Force Multiple Coils* | Force a set of consecutive coils ON or OFF |
| 10 | *Preset Multiple Registers* | Place specific binary values into a series of consecutive holding registers |
| 17 | *Report Slave ID* | Lets the master decide the slave type and status of the slave's RUN light |
| 18 ... FF | Reserved | N/A |

## F.3.1 Specifying Discrete and Register References

In the Modbus message, the type of reference to be accessed (e.g., discrete coil, discrete input, input register, or holding register) is specified in the function code. All discrete or register addresses within the Modbus data portion of the message are in hexadecimal, and are numbered relative to a base reference of zero. The address is specified as an offset from the base reference (0000) of that type.

For example, the first holding register in the 984 controller is termed 40001, and is addressed in a *Read Holding Registers* command as register address 0000. Similarly, the first coil is termed 00001 and is addressed in a *Read Coil Status* command as 0000. Coil 00127 is addressed as 007E hexadecimal (126 decimal).

☞ **Note:** All numbers in Modbus are in hexadecimal and are referred to in *high byte* and *low byte* format. In the above example, coil 00127 (007E hex) is represented with a high byte of 00 and a low byte of 7E.

## F.4 *Read Coil Status* (Function 01)

*Read Coil Status* reads the power bit of 0*x* coils and packs them together eight per byte, starting at the least significant bit of the first byte. Any unused bits in the last byte are padded with zeros at the high order end of the byte. The maximum number of coils that can be read is 2000, returned in up to 250 bytes.

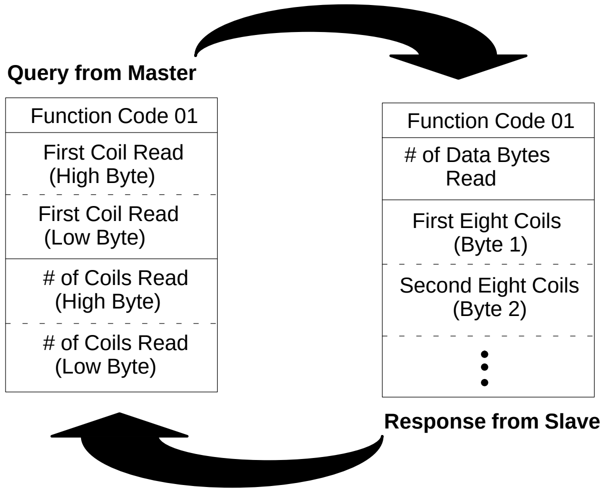

Query from Master
Function Code 01
First Coil Read (High Byte)
First Coil Read (Low Byte)
# of Coils Read (High Byte)
# of Coils Read (Low Byte)
Function Code 01
# of Data Bytes Read
First Eight Coils (Byte 1)
Second Eight Coils (Byte 2)
Response from Slave

**Figure 47 Read Coil Status**

☞ **Note:** All 0*x* coils specified in the query must exist in the PLC for this command to be processed. If not, the slave device will return an exception response.

## F.5 *Read Input Status* (Function 02)

*Read Input Status* reads the power bit of 1*xxxx* discrete inputs and packs them together eight per byte, starting at the least significant bit of the first byte. Any unused bits in the last byte are padded with zeros at the high order end of the byte. The maximum number of discrete inputs that can be read is 2000, returned in up to 250 bytes.

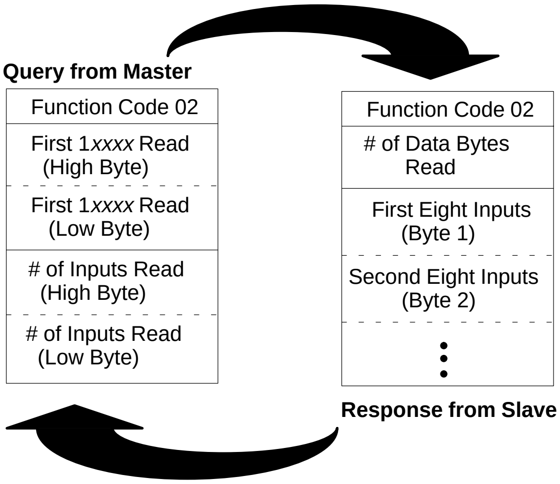
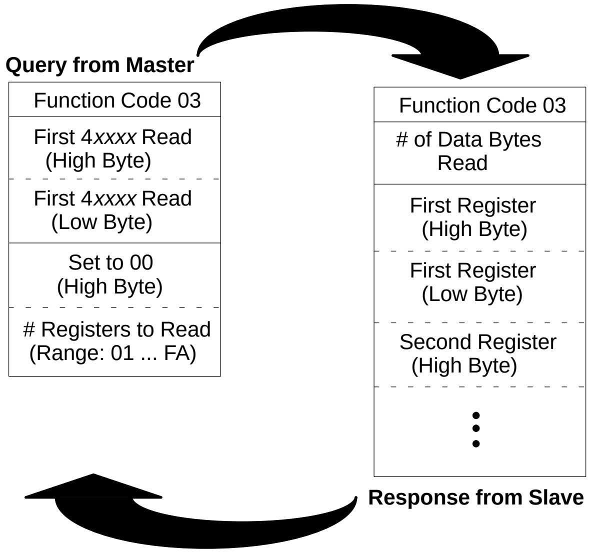

Query from Master
Function Code 02
First 1*xxxx* Read (High Byte)
First 1*xxxx* Read (Low Byte)
# of Inputs Read (High Byte)
# of Inputs Read (Low Byte)
Function Code 02
# of Data Bytes Read
First Eight Inputs (Byte 1)
Second Eight Inputs (Byte 2)
Response from Slave

**Figure 48 Read Input Status**

**Note:** All 1*x* inputs specified in the query must exist in the PLC for this command to be processed. If not, the slave device will return an exception response.

## F.6 *Read Holding Registers* (Function 03)

*Read Holding Registers* reads the contents of 4*xxxx* output registers and returns them two bytes per register. The maximum number of 4*xxxx* registers that can be read in one function 03 operation is 125, returned in up to 250 bytes.



Query from Master

- Function Code 03
- First 4*xxxx* Read (High Byte)
- First 4*xxxx* Read (Low Byte)
- Set to 00 (High Byte)
- # Registers to Read (Range: 01 ... FA)

Response from Slave

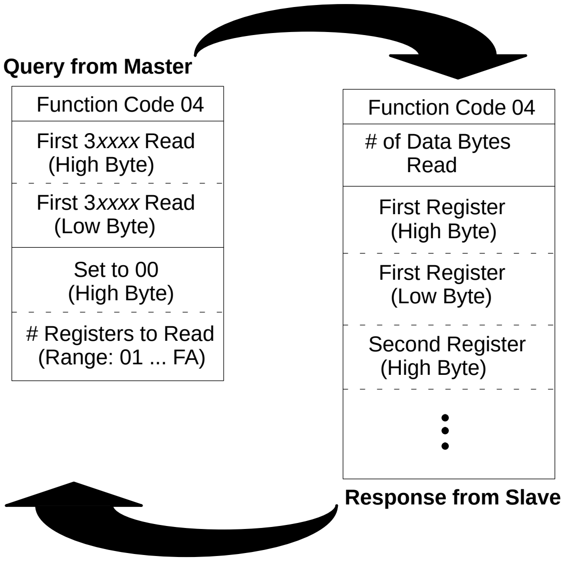
- Function Code 03
- # of Data Bytes Read
- First Register (High Byte)
- First Register (Low Byte)
- Second Register (High Byte)
- ⋮

**Figure 49 Read Holding Registers**

**Note:** All 4*x* registers specified in the query must exist in the PLC for this command to be processed. If not, the slave device will return an exception response.

## F.7 *Read Input Registers* (Function 04)

*Read Input Registers* reads the contents of 3*xxxx* input registers and returns them two bytes per register. The maximum number of 3*xxxx* registers that can be read in one function 04 operation is 125, returned in up to 250 bytes.



Query from Master
Function Code 04
First 3xxxx Read (High Byte)
First 3xxxx Read (Low Byte)
Set to 00 (High Byte)
# Registers to Read (Range: 01 ... FA)
Function Code 04
# of Data Bytes Read
First Register (High Byte)
First Register (Low Byte)
Second Register (High Byte)
Response from Slave

**Figure 50 Read Input Registers**

**Note:** All 3*x* registers specified in the query must exist in the PLC for this command to be processed. If not, the slave device will return an exception response.

## F.8 *Force Single Coil* (Function 05)

A *Force Single Coil* command sets or clears the current power state of a coil. The history bit associated with the coil is also updated so that transitionals will work correctly. The function uses four bytes for the query and four bytes for the response; the byte implementation is identical in both the query and response:

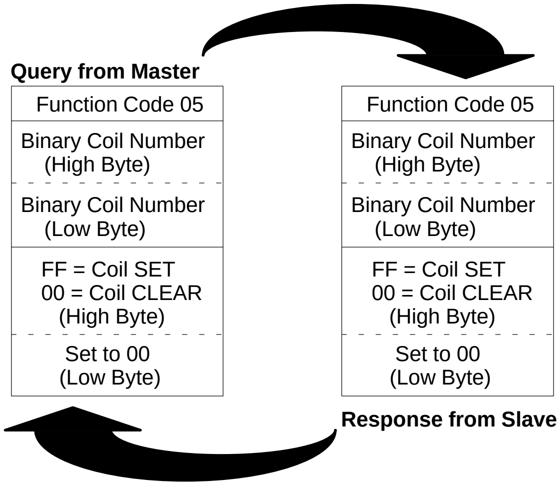


**Figure 51 Force Single Coil**

The first two bytes give you 16 bits with which to specify in binary notation the number of the coil to be forced. To force the coil ON, set all eight bits in the third byte to 1; to force the coil OFF, clear all eight bits in the third byte to 0. The fourth byte must be present, and its eight bits must always be cleared to 0.

☞ **Note:** Remember when you specify a coil number in bytes 1 and 2 that the binary representation is 1 less than the reference—e.g., coil 0 corresponds to reference 00001.

⚠ **Caution: Function 05 will override both the controller's MEMORY PROTECT and the coil DISABLE state.**

## F.9 *Preset Single Register* (Function 06)

A *Force Single Register* command sets the contents of a *4x* register. The function uses four bytes for the query and four bytes for the response; the byte implementation is identical in both the query and response:

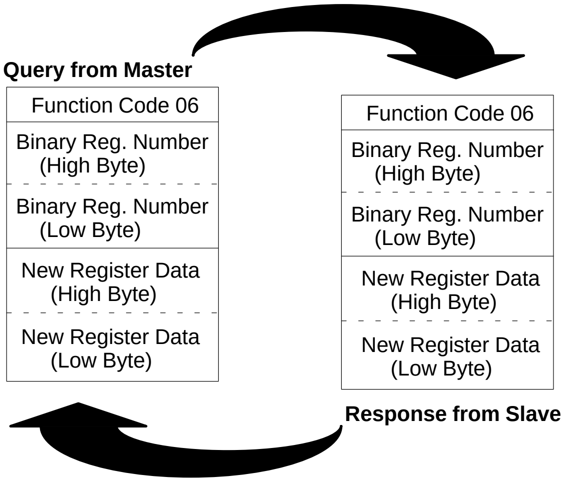

Query from Master
Function Code 06
Binary Reg. Number (High Byte)
Binary Reg. Number (Low Byte)
New Register Data (High Byte)
New Register Data (Low Byte)
Function Code 06
Binary Reg. Number (High Byte)
Binary Reg. Number (Low Byte)
New Register Data (High Byte)
New Register Data (Low Byte)
Response from Slave

**Figure 52 Preset Single Register**

The first two bytes give you 16 bits with which to specify in binary notation the number of the register to be set. The last two bytes give you 16 bits with which to specify the new register content.

☞ **Note:** Remember when you specify the register number in bytes 1 and 2 that the binary representation is 1 less than the reference - e.g., register 0 corresponds to reference 40001.

⚠ **Caution: Function 06 will override the controller's MEMORY PROTECT.**

## F.10 *Read Exception Status* (Function 07)

A *Read Exception Status* function requests the current power state of the first eight coils in the PLC. The state of these coils is returned in a single data byte in the response—the power state of the first coil is returned in the least significant bit of the data byte.

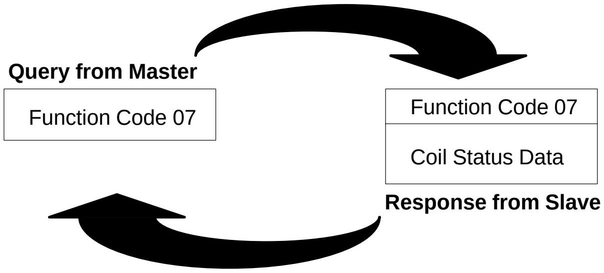


**Figure 53 Read Exception Status**

# F.11 *Get/Clear Network Statistics* (Function 08)

In order to obtain network statistics from a host device, function 08 must be used in conjunction with subfunction code 21 (hex 15), which is specified in the first two bytes following the function code in the query. To *Get Statistics*, specify 03 in the fourth byte following the function code; to *Clear Statistics*, specify 04 in that byte.

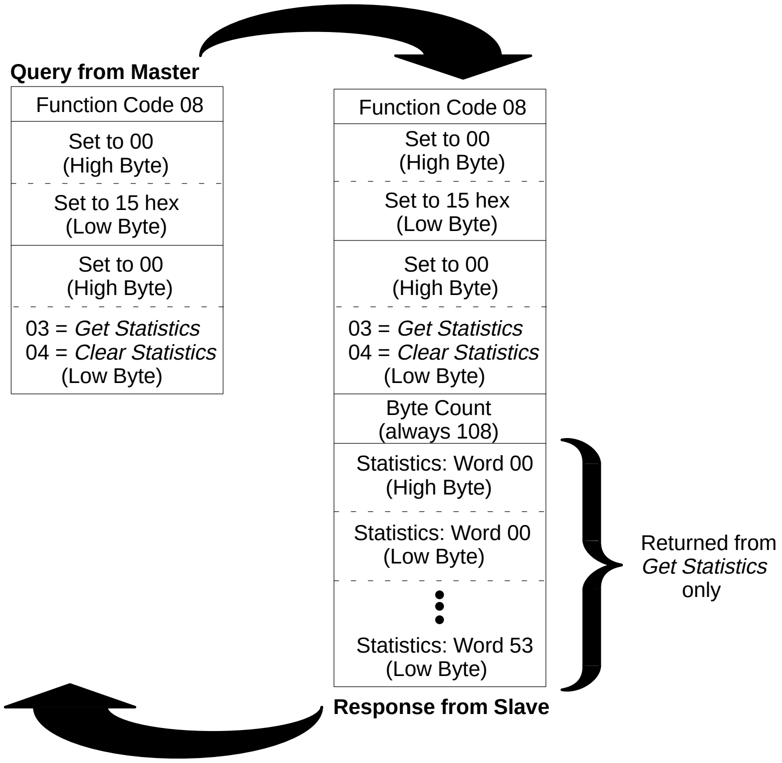
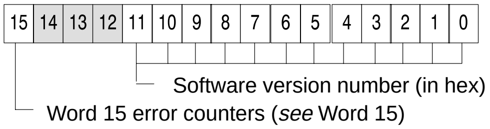


**Figure 54 Get/Clear Network Statistics**

**Note:** Other diagnostic subfunctions accessible through Modbus function 08 are not applicable to Modbus Plus networks.

The Modbus Plus network statistics accessible via this function are described on the next four pages.

## F.11.1 Modbus Plus Network Statistics

| Word | | Meaning |
|---|---|---|
| 00 | **Bit** | Node type ID |
| | 0 | Unknown node type |
| | 1 | PLC node |
| | 2 | Modbus bridge node |
| | 3 | Host computer node |
| | 4 | Bridge Plus node |
| | 5 | Peer I/O node |
| | 6 | reserved |
| | 7 | 1 = Supports Timer pre–scaling, and Timer has pre–scale value<br>0 = Does not support Timer pre–scaling, or does not have value |
| 01 | **Bit** | |
| | 0 ... 11 | Software version number in hex (to read, strip bits 12–15 from word) |
| | 12 | Device supports dual cable network |
| | 13 | Device supports Peer Cop communication |
| | 14 | Device supports identity reporting |
| | 15 | Defines Word 15 error counters (see Word 15)<br>Most significant bit defines use of error counters in Word 15. Least significant half of upper byte, plus lower byte, contain software version.<br> 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0<br>Software version number (in hex)<br>Word 15 error counters (*see* Word 15) |
| 02 | | Network address for this station |
| 03 | **Bit** | MAC state variable: |
| | 0 | Power up state |
| | 1 | Monitor offline state |
| | 2 | Duplicate offline state |
| | 3 | Idle state |
| | 4 | Use token state |
| | 5 | Work response state |
| | 6 | Pass token state |
| | 7 | Solicit response state |
| | 8 | Check pass state |
| | 9 | Claim token state |
| | 10 | Claim response state |

| Word | | Meaning |
|---|---|---|
| 04 | | Peer status (LED code); provides status of this unit relative to the network: |
| | 0 | Monitoring link operation only—passive station |
| | 32 | Normal link operation |
| | 64 | Never getting token—sees tokens, receives none |
| | 96 | Sole station—never sees tokens |
| | 128 | Duplicate station—sees other stations with same address |
| 05 | | Token pass counter; increments each time this station gets the token |
| 06 | | Token rotation time in ms |
| 07 | **Byte** | |
| | LO | Data master failed during token ownership bit map |
| | HI | Program master failed during token ownership bit map |

☞ **Note:** Word 07 bitmaps are used internally by the peer processor to determine which paths have already had a command sent to them during the current token ownership. This limits the number of commands per path to one during a single token ownership.

| Word | Byte | Meaning |
|---|---|---|
| 08 | LO | Data master token owner work-to-do table |
| | HI | Program master token owner work-to-do table |
| 09 | LO | Data slave token owner work-to-do table |
| | HI | Program slave token owner work-to-do table |
| 10 | LO | Data master response (now available to read) |
| | HI | Data slave command |
| 11 | LO | Program master response (now available to read) |
| | HI | Program slave command |
| 12 | LO | Program master connect status table—master paths in use |
| | HI | Program slave automatic logout request table—slaves to log out |

☞ **Note:** Words 08 ... 12 are token owner work tables. They are bitmaps representing work that needs to be done by the node the next time it gets the token. Each byte is a bitmap corresponding to work requested of each of the eight paths of the indicated type.

| Word | Byte | Meaning |
|---|---|---|
| 13 | LO | Pretransmit deferral error counter |
|  | HI | Receive buffer DMA overrun error counter |
| 14 | LO | Repeated command received counter |
|  | HI | Frame size error counter |
| 15 |  | If Word 1 bit 15 is *not set*, Word 15 has the following meaning: |
|  | LO | Receiver collision-abort error counter |
|  | HI | Receiver alignment error counter |
|  |  | **Note** If Word 1 bit 15 is *set*, Word 15 has the following meaning: |
|  | LO | Cable A framing error |
|  | HI | Cable B framing error |
| 16 | LO | Receiver CRC error counter |
|  | HI | Bad packet-length error counter |
| 17 | LO | Bad link-address error counter |
|  | HI | Transmit buffer DMA-underrun error counter |
| 18 | LO | Bad internal packet length error counter |
|  | HI | Bad MAC function code error counter |
| 19 | LO | Communication retry counter |
|  | HI | Communication failed error counter |
| 20 | LO | Good receive packet success counter (increments normally) |
|  | HI | No response received error counter (increments normally 1 ... 10 times/s). Each station occasionally allows a new station to join the network, which increments this counter. If a station leaves, the remaining stations continue to increment their error counters until the station is removed from each station's map. |
| 21 | LO | Exception response received error counter—LLC layer error, illegal packet error |
|  | HI | Unexpected path error counter—data packet contains illegal path field |
| 22 | LO | Unexpected response counter—packet sent to wrong destination |
|  | HI | Forgotten transaction error counter—command was initiated but never completed, possibly because the response packet had the wrong path, sequence numbers, or node number. |

**Note:** Words 13 ... 22 contain pairs of 8-bit counters that pertain to certain types of error conditions as well as to successful transactions. Under normal operating conditions, the only bytes that change are word 20 LO and HI. Word 14 HI could also increment because of an MSTR or similar programming error in the application. If any other bytes increments, a possible problem exists on the network—e.g., in a single station or wiring connection.

| Word | Byte | Meaning |
|---|---|---|
| 23 | LO | Active station table bit map, nodes 8 ... 1 |
|  | HI | Active station table bit map, nodes 16 ...9 |
| 24 | LO | Active station table bit map, nodes 24 ... 17 |
|  | HI | Active station table bit map, nodes 32 ... 25 |
| 25 | LO | Active station table bit map, nodes 40 ... 33 |
|  | HI | Active station table bit map, nodes 48 ... 41 |
| 26 | LO | Active station table bit map, nodes 56 ... 49 |
|  | HI | Active station table bit map, nodes 64 ... 57 |

☞ **Note:** Words 23 ... 26 contain the active station bitmaps. An active station is any one that has sent packets of data over the network.

| Word | Byte | Meaning |
|---|---|---|
| 27 | LO | Token station table bit map, nodes 8 ... 1 |
|  | HI | Token station table bit map, nodes 16 ...9 |
| 28 | LO | Token station table bit map, nodes 24 ... 17 |
|  | HI | Token station table bit map, nodes 32 ... 25 |
| 29 | LO | Token station table bit map, nodes 40 ... 33 |
|  | HI | Token station table bit map, nodes 48 ... 41 |
| 30 | LO | Token station table bit map, nodes 56 ... 49 |
|  | HI | Token station table bit map, nodes 64 ... 57 |

☞ **Note:** Words 27 ... 30 contain the token station table bitmaps. A token station is any one that has token-passing capabilities.

| Word | Byte | Meaning |
|---|---|---|
| 31 | LO | Global data present table bit map, nodes 8 ... 1 |
|  | HI | Global data present table bit map, nodes 16 ...9 |
| 32 | LO | Global data present table bit map, nodes 24 ... 17 |
|  | HI | Global data present table bit map, nodes 32 ... 25 |
| 33 | LO | Global data present table bit map, nodes 40 ... 33 |
|  | HI | Global data present table bit map, nodes 48 ... 41 |
| 34 | LO | Global data present table map, nodes 56 ... 49 |
|  | HI | Global data present table bit map, nodes 64 ... 57 |

**Note:** Words 31 ... 34 contain the global data present table bitmaps. Each time a station passes a token, it also passes the global data, even if there are zero bytes of global data. When one station sees another pass the token with global data, it sets its bit in its table for that other station. The bit remains set until the station reads the global data from that other station, after which the bit is cleared. A second read of global data indicates that no global data is present.

**Note:** In screen 2 of the MBPSTAT program, the number of global data words present is indicated under the station number. If this field is filled with spaces, then MBPSTAT has requested the global data from a second time before the other station passed the token.

| Word | Byte | Meaning |
|---|---|---|
| 35 | LO | Receive buffer in use bit map, buffer 8 ... 1 |
| | HI | Receive buffer in use bit map, buffer 16 ... 9 |
| 36 | LO | Receive buffer in use bit map, buffer 24 ... 17 |
| | HI | Receive buffer in use bit map, buffer 32 ... 25 |
| 37 | LO | Receive buffer in use bit map, buffer 40 ... 33 |
| | HI | Station management command processed initiation counter |

**Note:** The LO bytes of words 35 ... 37 indicate the use of the internal receive buffers within the peer processor.

| Word | Byte | Meaning |
|---|---|---|
| 38 | LO | Data master output path 1 command initiation counter |
| | HI | Data master output path 2 command initiation counter |
| 39 | LO | Data master output path 3 command initiation counter |
| | HI | Data master output path 4 command initiation counter |
| 40 | LO | Data master output path 5 command initiation counter |
| | HI | Data master output path 6 command initiation counter |
| 41 | LO | Data master output path 7 command initiation counter |
| | HI | Data master output path 8 command initiation counter |
| 42 | LO | Data slave input path 41 command processed counter |
| | HI | Data slave input path 42 command processed counter |
| 43 | LO | Data slave input path 43 command processed counter |
| | HI | Data slave input path 44 command processed counter |
| 44 | LO | Data slave input path 45 command processed counter |
| | HI | Data slave input path 46 command processed counter |
| 45 | LO | Data slave input path 47 command processed counter |

| Word | Byte | Meaning |
|---|---|---|
| | HI | Data slave input path 48 command processed counter |
| 46 | LO | Program master output path 81 command initiation counter |
| | HI | Program master output path 82 command initiation counter |
| 47 | LO | Program master output path 83 command initiation counter |
| | HI | Program master output path 84 command initiation counter |
| 48 | LO | Program master command initiation counter |
| | HI | Program master output path 86 command initiation counter |
| 49 | LO | Program master output path 87 command initiation counter |
| | HI | Program master output path 88 command initiation counter |
| 50 | LO | Program slave input path C1 command processed counter |
| | HI | Program slave input path C2 command processed counter |
| 51 | LO | Program slave input path C3 command processed counter |
| | HI | Program slave input path C4 command processed counter |
| 52 | LO | Program slave input path C5 command processed counter |
| | HI | Program slave input path C6 command processed counter |
| 53 | LO | Program slave input path C7 command processed counter |
| | HI | Program slave input path C8 command processed counter |

**Note:** Station management commands (the Get Statistics command) are trapped by the peer processor, which formats the response message and sends it to the requestor. This does not consume any of the internal paths.

The number of station management commands received from the network is reflected in the 8-bit counter of word 37 LO. This counter increments with station management commands received over the network. It does not increment if a command is received locally—e.g., if node 20 requests its own statistics—and therefore consumes no network traffic.

Each type of path has a predefined numerical value as shown in the words above. The (hex) values are:

□ Data Master: 1 ... 8

□ Data Slave: 41 ... 48

□ Program Master: 81 ... 88

□ Program Slave: C1 ... C8

Transaction counters are maintained for each path in words 38 ... 53. (A total of 32 8-bit counters exists, two counters per word.)

Each time a command is processed—i.e., sent for a Master path or received for a Slave path—the counter increments. When the count reaches 255 (decimal), it wraps to zero.

The MBPSTAT program displays the information in screen 9. It shows both the actual and the total value. The *actual value* is the current counter value returned by the Get Statistics command. The *total value* is the accumulated total count since screen 9 was last active.

## F.12 *Force Multiple Coils* (Function 0F)

A *Force Multiple Coils* function sets or clears the current power state of a set of up to 800 consecutive coils. The history bits associated with the coils are also updated so that transitionals will work correctly.

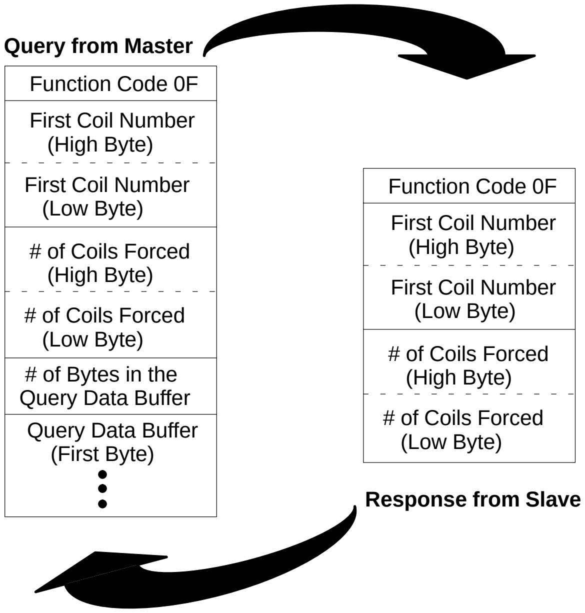

Query from Master
Function Code 0F
First Coil Number (High Byte)
First Coil Number (Low Byte)
# of Coils Forced (High Byte)
# of Coils Forced (Low Byte)
# of Bytes in the Query Data Buffer
Query Data Buffer (First Byte)
Function Code 0F
First Coil Number (High Byte)
First Coil Number (Low Byte)
# of Coils Forced (High Byte)
# of Coils Forced (Low Byte)
Response from Slave

**Figure 55 Force Multiple Coils**

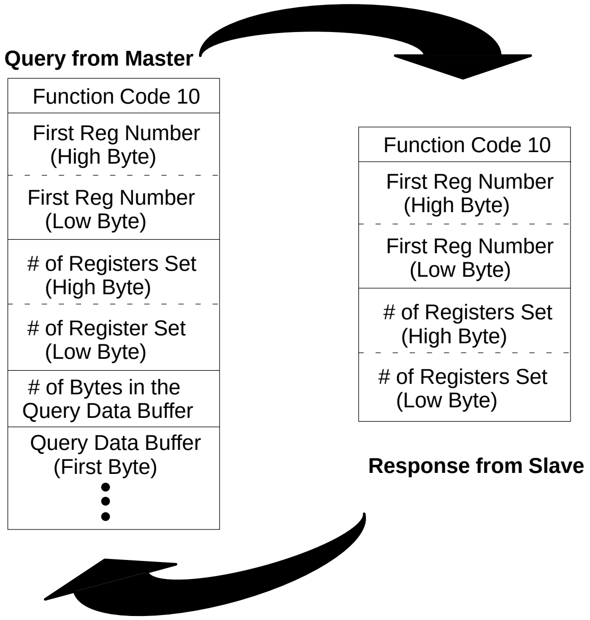
The number of bytes in the query data buffer is the integer part of (*# of Coils Forced* + 7) divided by 8. A byte in this data buffer stores the state of eight consecutive coils. To force a coil ON, set its representative bit to 1; to force the coil OFF, clear the associated bit to 0.

☞ **Note:** Remember when you specify coil references in the data bytes that the binary representation is 1 less than the coil reference number—e.g., coil 0 in the data corresponds to reference 00001.

⚠ **Caution: Function 0F will override both the controller's MEMORY PROTECT and the coil DISABLE state.**

## F.13 *Preset Multiple Registers* (Function 10 hex)

A *Preset Multiple Registers* function presets the contents of a group of up to 100 consecutive 4*xxxx* registers, filling two bytes per register.



Query from Master

Function Code 10<br>First Reg Number (High Byte)<br>First Reg Number (Low Byte)<br># of Registers Set (High Byte)<br># of Register Set (Low Byte)<br># of Bytes in the Query Data Buffer<br>Query Data Buffer (First Byte)<br>⋮

Response from Slave

Function Code 10<br>First Reg Number (High Byte)<br>First Reg Number (Low Byte)<br># of Registers Set (High Byte)<br># of Registers Set (Low Byte)

**Figure 56 Preset Multiple Registers**

The number of bytes in the query data buffer is the number of 4*xxxx* registers to be preset x 2. The high and low order data bytes will be placed into two consecutive bytes in the destination register.

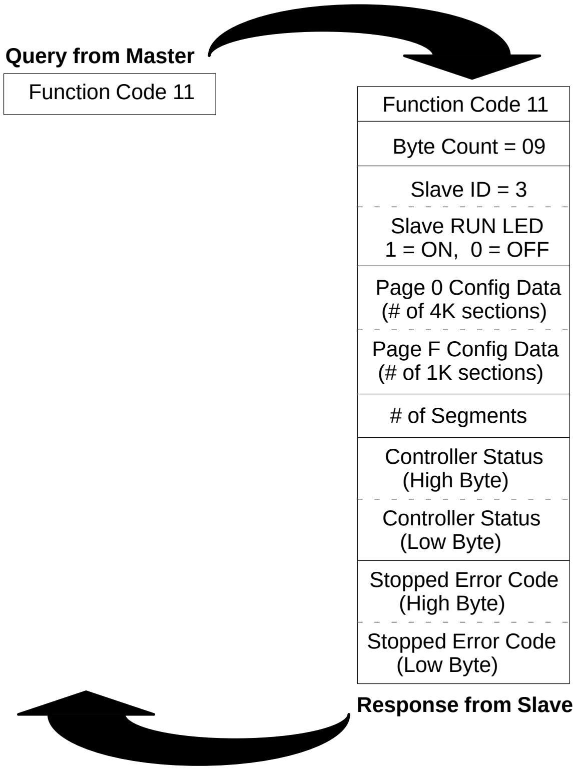
For example, if you specify a value of 50 bytes, you will be filling a total of 25 registers.

**Note:** When you specify a register in the First Register field, the binary representation is 1 less than the reference number — e.g., specifying register 0 in the field will address reference 40001.

**Caution: Function 10 hex will override the controller's MEMORY PROTECT.**

# F.14 *Report Slave ID* (Function 11 hex)

Issuing a *Report Slave ID* function lets you obtain in a single response the slave type, its RUN light state, configuration information, machine status, and any current stopped error code.



Query from Master
Function Code 11
Function Code 11
Byte Count = 09
Slave ID = 3
Slave RUN LED 1 = ON, 0 = OFF
Page 0 Config Data (# of 4K sections)
Page F Config Data (# of 1K sections)
# of Segments
Controller Status (High Byte)
Controller Status (Low Byte)
Stopped Error Code (High Byte)
Stopped Error Code (Low Byte)
Response from Slave

**Figure 57 Report Slave ID**

### F.14.1 Reading the Controller Status

Word 101 in the 984 configuration table (in page 0 of system memory) indicates certain states of the controller. A state may be valid for the life of the controller (for example, the size of the configuration), or may be set conditionally by external events (for example, MEMORY PROTECT currently ON or OFF).

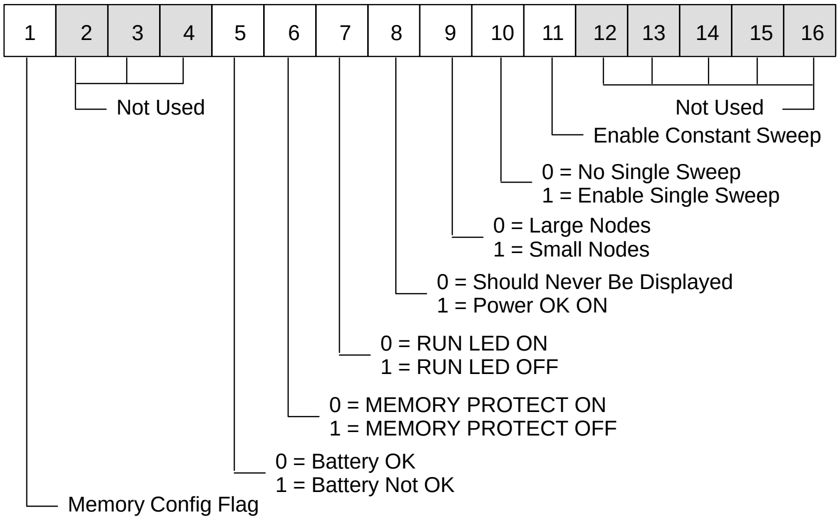


**Figure 58 Reading Controller Status**

Information about the PLC’s stopped error codes can also be obtained with this command. The codes are described on the next page.

## F.14.2 Reading Stopped Error Codes

Stopped error codes for any 984 programmable controller on a Modbus Plus network can be accessed via function 11 hex. The current stopped state condition, if one exists, is stored in word 105 in the 984 configuration table (in page 0 of system memory):

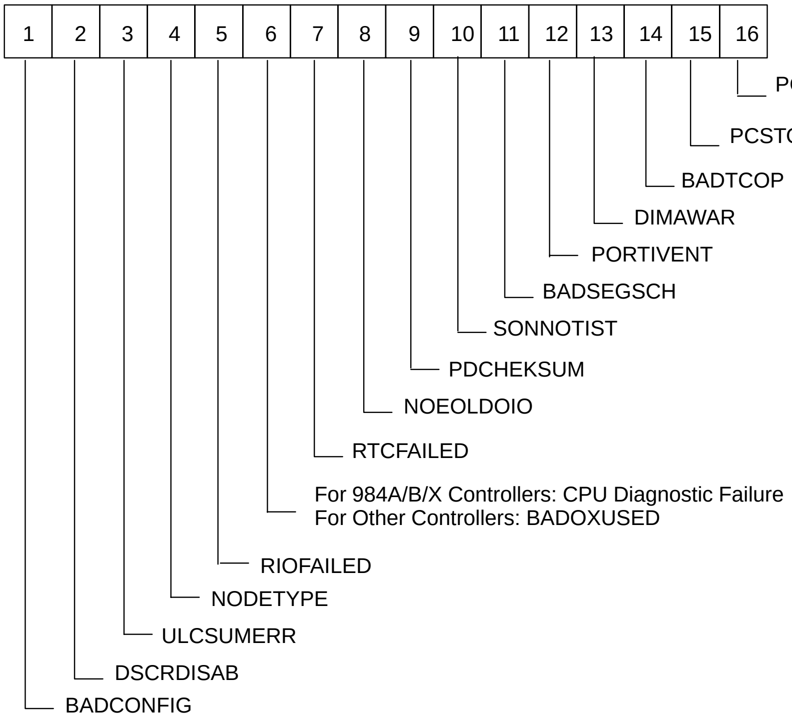


**Figure 59 Reading Stopped Error Codes**

| **Legend** | **Meaning** |
|---|---|
| PCSICK | Controller unhealthy |
| PCSTOPPED | Controller stopped |
| BADTCOP | Bad I/O traffic cop table |
| DIMAWAR | PLC in DIM AWARENESS state |
| PORTIVENT | Bad port intervention |
| BADSEGSCH | Bad segment scheduler |
| SONNOTIST | Start of network (SON) did not start segment |
| PDCHEKSUM | Bad power–down checksum |
| NOEOLDOIO | Watchdog timer has expired |
| RTCFAILED | Real time clock failure |
| BADOXUSED | Bad *coil used* table |
| RIOFAILED | Remote I/O failure |
| NODETYPE | Illegal node type used |
| ULCSUMERR | User logic checksum error |
| DSCRDISAB | Discrete disable error |
| BADCONFIG | Bad configuration table |

# F.15 Exception Responses

When a programming or operation error occurs while issuing any of these Modbus data access commands, the slave responds to the master with an error message in the form of an *exception code*. Typical errors that elicit exception responses include illegal data in a message, a controller's failure to respond to an interface panel, or a difficulty communicating with a slave.

Four exception responses are possible in a Modbus Plus data acquisition operation:

| Exception Code | Error Condition |
|---|---|
| 01 | Illegal function for the addressed slave |
| 02 | Illegal data address within the information field for the addresses slave |
| 03 | Illegal data value in the information field for the addressed slave |
| 06 | Busy—the function just requested cannot be performed at this time because a long–duration PROGRAM command is being processed; reissue the command later |

Exception codes 04, 05, and 07 are not applicable in Modbus Plus applications, and exception codes 08 ... FF are reserved.

When a slave detects an error, it sends a response message to the master consisting of a function code and one of the above exception codes. The function code associated with an exception response is the original code plus 80 hexadecimal — i.e., the high order bit in the function code of the original query is set to 1.

## F.15.1 An Example

Here is an example of an incorrect query and the subsequent exception response. The query is:

| Function | Start Address (High Byte) | Start Address (Low Byte) | # of Coils (High Byte) | # of Coils (Low Byte) |
|---|---|---|---|---|
| 01 | 04 | A1 | 00 | 01 |

**Figure 60 Exception: Query Example**

The query above is a *Read Coil Status*, requesting the status of coil 1245 decimal. If the controller is a 1 K machine, this is an invalid reference. Consequently, the following exception response is generated:

| Function Code | Exception Code |
|---|---|
| 81 | 02 |

**Figure 61 Exception: Response Example**

The function code is the original query code plus 80 hexadecimal (its high order bit is set to 1). Exception code 02 indicates an illegal data address.

## F.15.2 Responding as a Slave

A Modbus Plus application that handles Modbus protocol must function as a slave. Other devices on the network will function as masters, sending queries to the application. The application can determine how to respond to those queries.

The application should issue an exception response for all Modbus function codes that it does not recognize. Otherwise, the application should create the correct response message for the selected Modbus function code.

# Appendix G. Modbus Plus Statistics Explained

# G.1 Modbus Plus Statistics Explained

This Appendix provides practical information about Modbus Plus Network Statistics. It presents a detailed description of what each word contains and how it can be used. This information should prove helpful during the startup and troubleshooting of a Modbus Plus Network.

This information is displayed either when using the MBPSTAT program, or when programming an MSTR instruction block with the Get Remote Statistics Function Code 8.

The MBPSTAT program would be used when the user is troubleshooting a Modbus Plus network. The program shows the statistics information in a series of screens.

The MSTR block would be used to enable the user application program to access statistics information and use it in the ladder logic. Using the MSTR block with function code 8 will return 54 words of information, ranging from Word 0 to Word 53.

## G.1.1 Statistics Data Layout

The following table shows the Node Error Statistics reported by MBPSTAT and their corresponding words returned by the MSTR block.

| MBPSTAT Node Error Statistic | MSTR Word, Byte |
|---|---|
| Pre–transmit deferral error counter | Word 13, LO byte |
| Receive buffer DMA overrun error counter | Word 13, HI byte |
| Repeated Command received error counter | Word 14, LO byte |
| No Try (nonexistent station) error counter | Word 14, HI byte |
| Cable A framing error | Word 15, LO byte (Word 1, bit 15 = 1) |
| Cable B framing error | Word 15, HI byte (Word 1, bit 15 = 1) |
| Receiver CRC error counter | Word 16, LO byte |
| Bad packet length error counter | Word 16, HI byte |
| Transmit buffer DMA underrun error counter | Word 17, HI byte |
| Bad internal packet–length error counter | Word 18, LO byte |
| Bad MAC–function–code error counter | Word 18, HI byte |
| Communication failed error counter | Word 19, HI byte |
| Good receive packet success counter | Word 20, LO byte |
| No response received error counter | Word 20, HI byte |
| Exception response received error counter | Word 21, LO byte |
| Unexpected path error counter | Word 21, HI byte |
| Unexpected response error counter | Word 22, LO byte |
| Forgotten transaction error counter | Word 22, HI byte |

### G.1.2 Word 0

| Word | Bits | Meaning |
|---|---|---|
| 0 | | Node Type |
| | 1 | Unknown Node Type |
| | 2 | PLC Node |
| | 3 | Host Computer Node |
| | 4 | Bridge Plus Node |
| | 5 | Peer I/O Node |

Word 0 is the Node Type word. The bit that is set reflects what kind of device is present at the network node. When you issue a command to get information from a node, this word tells you what type of node it is.

### G.1.3 Word 1

| Word | Bits | Meaning |
|---|---|---|
| 1 | 0..11 | Software Version Number in hex |
| | 12..14 | Reserved |
| | 15 | Defines Word 15 Counter |

If you remove bits 12 through 15 and look at bits 0 to 11 you will receive the Software Version number in Hex. The software version number is the Firmware Code revision number of the peer processor in the CPU. If Bit 15 is on it shows that there is an error. See word 15 for the error description.

### G.1.4 Word 2

| Word | Meaning |
|---|---|
| 2 | Network Address for this station |

This word contains the Modbus Plus Address for the requested station, from 1 to 64.

## G.1.5 Word 3

| Word | Bits | Meaning |
|---|---|---|
| 3 | | MAC State Variable |
| | 0 | Power Up State |
| | 1 | Monitor Offline State |
| | 2 | Duplicate Offline States |
| | 3 | Idle State |
| | 4 | Use Token State |
| | 5 | Work Response State |
| | 6 | Pass Token State |
| | 7 | Solicit Response State |
| | 8 | Check Pass State |
| | 9 | Claim Token State |
| | 10 | Claim Response State |

Word 3 shows the status of the node in the Network. The MAC variables are variables that are used in the Modbus Plus Network. When these bits are set they imply what the node on the network is doing, for example Passing the Token or Claiming the Token. The MAC descriptions and details are in-depth to the design of the Modbus Plus Network and to the way the token rotation works.

## G.1.6 Word 4

| Word | Value | Meaning |
|---|---|---|
| 4 | | Peer Status (LED Code) provides status of this unit relative to the network. Its contents are: |
| | 0 | Monitor Link Operation |
| | 32 | Normal Link Operation |
| | 64 | Never Getting Token |
| | 96 | Sole Station |
| | 128 | Duplicate station |

The value contained in the word shows the status of the node on the Modbus Plus network.

Monitor Link Operation means that this node is monitoring the network for 5 seconds as it prepares to enter the normal token passing on the Modbus Plus network. This condition occurs directly after powerup or this node is no longer a duplicate station, because the other duplicate station was recently removed from the network.

Normal Link Operation means that this node has successfully integrated itself into the network token passing.

Never Getting Token means that this node hears other nodes on the network but no other node ever passes the token to this node. This is an error condition that most likely indicates a wiring or termination problem. In rare cases it could indicate that this node has a broken transmitter.

Sole Station means that this node is alone on the Modbus Plus network. This is an error condition only if this node is really connected to a Modbus Plus network that has other nodes present. This could be caused by a wiring or termination problem.

Duplicate Station means that there are two or more nodes on the Modbus Plus network with the same address. The station blinking duplicate station in its LED is not part of the token passing. The node monitors the network and remains offline as long as any packets from the active station, using this node address, are received within the last 5 seconds.

## G.1.7 Word 5

| **Word** | **Meaning** |
|---|---|
| 5 | Token pass counter; increments each time this station gets the token. |

This keeps track of how many times the token has rotated around in the network. Each time this station gets the token this number will increase by one.

## G.1.8 Word 6

| **Word** | **Meaning** |
|---|---|
| 6 | Token rotation time in milliseconds. |

This is the time in milliseconds it takes for one complete token rotation.

## G.1.9 Word 7 ... 12

Words 7 through 12 are used to coordinate the receive buffers which are in Words 35 to 37. These are really internal to the Peer Processor on the PLC and are not really needed for debugging the Modbus Plus Network.

| Word | Value | Meaning |
|---|---|---|
| 7 | LO Byte | Data Master Failed during token ownership bit map |
| | HI Byte | Program Master Failed during token ownership bit map |

| Word | Value | Meaning |
|---|---|---|
| 8 | LO Byte | Data Master token owner work bit map |
| | HI Byte | Program Master token owner work bit map |

| Word | Value | Meaning |
|---|---|---|
| 9 | LO Byte | Data Slave token owner work bit map |
| | HI Byte | Program Slave token owner work bit map |

| Word | Value | Meaning |
|---|---|---|
| 10 | LO Byte | Data Master response is now available to read |
| | HI Byte | Data Slave / Get Slave command transfer request bit map |

| Word | Value | Meaning |
|---|---|---|
| 11 | LO Byte | Program Master response is now available to read |
| | HI Byte | Program Slave/ Get Slave command transfer request bit map |

| Word | Value | Meaning |
|---|---|---|
| 12 | LO Byte | Program Master connect status bit map |
| | HI Byte | Program Slave automatic logout request bit map |

## G.1.10 Word 13

| Word | Value | Meaning |
|---|---|---|
| 13 | LO Byte | Pretransmit deferral error counter |
| | HI Byte | Receive DMA overrun error counter |

On the Modbus Plus network when a message is received there is a network silence that ensues. This network silence is there so that it will give this node time to construct a reply to the received message. If this node receives another message before it has sent out its reply there will be a Pretransmit deferral error. This can only occur with a MAC Layer problem which is typically wiring related.

Receive DMA Overrun error indicates that a Modbus Plus message was received that was too big. If this error is displayed, then please contact Schneider Automation technical support.

### G.1.11 Word 14

| Word | Value | Meaning |
|---|---|---|
| 14 | LO Byte | Repeated command received counter |
| | HI Byte | Destination node absent error counter |

Repeated command error can occur because of bad wiring and/or bad termination in the Modbus Plus network. This error will accumulate when a node is trying to pass the token and the acknowledgement is not received. After this the node will try to pass the token again.

Destination node absent error counter increments when this node owns the token and attempts to deliver a command to the destination node, but the destination node is not present in the active station table. Since the destination node is not present on the network, the command is not sent on the network, and a routing failure is generated. This improves network performance by avoiding the command packet time and the very likely no response delay timeout. The normal MAC layer processing must first find the destination node before any communication attempts to it will be made.

### G.1.12 Word 15

| Word | Value | Meaning |
|---|---|---|
| 15 | If Word 1 Bit 15 is not set, Word 15 has the following meanings: | |
| | LO Byte | Receiver Collision error counter |
| | HI Byte | Receiver Alignment error counter |
| 15 | If Word 1 Bit 15 is set , Word 15 has the following meanings; | |
| | LO Byte | Cable A Framing error |
| | HI Byte | Cable B Framing error |

Receiver Collision or Alignment errors occur when the network is not properly wired or terminated.

Framing errors are if the frame of the Modbus Plus message is incorrect in its form. The frame of the message is garbled. This is typically caused by bad wiring or incorrect termination of the Modbus Plus network wiring.

## G.1.13 Word 16

| Word | Value | Meaning |
|---|---|---|
| 16 | LO Byte | Receiver CRC error counter |
| | HI Byte | Bad Packet length error counter |

When the Modbus Plus message is generated there is a CRC, or cyclic redundancy check. This is a binary encoded number that is unique to the data contained in the actual packet. If the CRC number received by the PLC is not correct according to the data that was received, then the message is considered garbled and invalid. This is typically caused by bad wiring or incorrect termination of the Modbus Plus network wiring.

Bad Packet length error counter occurs when the number of packet bytes actually received is different from the internal packet length field contained in the received packet. If this error is displayed, then please contact Schneider Automation technical support.

## G.1.14 Word 17

| Word | Value | Meaning |
|---|---|---|
| 17 | LO Byte | Bad link address error counter |
| | HI Byte | Transmit buffer DMA underrun error counter |

Bad link address error counter occurs when the source address or destination address contained in a received packet is not 1 through 64. If this error is displayed, then please contact Schneider Automation technical support.

Transmit buffer DMA underrun error occurs when the LAN controller has completed transmission of a packet, but the DMA controller has more packet bytes to move to the LAN. If this error is displayed, then please contact Schneider Automation technical support.

### G.1.15 Word 18

| Word | Value | Meaning |
|---|---|---|
| 18 | LO Byte | Bad internal packet length error counter |
| | HI Byte | Bad MAC function code error counter |

Bad internal packet length error counter occurs when a received packet has a total length that is incorrect for the function code. If this error is displayed, then please contact Schneider Automation technical support.

Bad MAC function code error counter occurs when a received packet contains an unsupported MAC function code. If this error is displayed, then please contact Schneider Automation technical support.

### G.1.16 Word 19

| Word | Value | Meaning |
|---|---|---|
| 19 | LO Byte | Communication retry error counter |
| | HI Byte | Communication failed error counter |

Communication retry occurs when the token owner sends a command packet to a destination node that is presumed to be present on the network, and no response packet is received within the timeout period. The retry counter is incremented and the next attempt is made. This error indicates a wiring or termination problem.

Communication failed error occurs when the token owner fails to receive a response after the initial command attempt and all subsequent retries have been exhausted. This error indicates a wiring or termination problem.

## G.1.17 Word 20

| Word | Value | Meaning |
|---|---|---|
| 20 | LO Byte | Good receive packet success counter |
|  | HI Byte | No response received error counter |

Good receive packet success counter should be incrementing normally in a good Modbus Plus Network. This indicates a normally operating Modbus Plus Network. If this does not increment or if it increments slowly this would indicate problems in the network, probably wiring.

No response received error increments every time that the token owner has sent a point to point related command packet to a destination node, and no response packet was received within the response timeout period. This can occur during normal operation because the MAC layer makes frequent attempts to locate previously non-existent nodes that might have just been added to the network. If the destination node is still not present, the usual case, then there is a no response error. This error will also occur when wiring errors are present.

## G.1.18 Word 21

| Word | Value | Meaning |
|---|---|---|
| 21 | LO Byte | Exception response received error counter |
|  | HI Byte | Unexpected path error counter |

Exception response received occurs when the token owner sends a command packet to destination node and an exception response is received. This usually indicates that the destination node had previously initiated a point to point command and then subsequently aborted it.

Unexpected path error counter occurs when the token owner sends a point to point related command packet to destination and a response is received that has the wrong path number. If this error is displayed, then please contact Schneider Automation technical support.

### G.1.19 Word 22

| Word | Value | Meaning |
|---|---|---|
| 22 | LO Byte | Unexpected response error counter |
| | HI Byte | Forgotten transaction error counter |

Unexpected response occurs when the token owner sends a point to point related command packet to a destination and a totally unexpected response is received.  If this error is displayed, then please contact Schneider Automation technical support.

Forgotten transaction occurs under the following conditions:   When the token owner has successfully initiated a MODBUS command on a point to point master path and has waited 10 seconds without receiving anything from the destination node concerning this transaction, then the token owner requests transaction status from the destination node. If the destination node does not remember this transaction, then the token owner node aborts the transaction and increments the counter. This error might occur if a third party driver at the destination node already delivered the response incorrectly to the wrong node.

### G.1.20 Word 23 ... 37

Words 23 to 37 represent table bit maps. A table bit map is a series of words in which the bits in each word represent one item's state. For example, Word 23 has 16 bits total. Eight bits are in the LO Byte and eight bits are in the HI byte. Word 23 LO Byte show whether nodes 1 to 8 are present on the Modbus Plus Network. The HI Byte shows the presence of nodes 9 to 16.

The Active Station table bit map (Words 23 ... 26) is 64 bits that are dedicated in each station in the Modbus Plus Network. Each bit in the map signifies one individual station. When a bit is set, it signifies that the node is active on the Modbus Plus Network. Each station must know the active nodes that are available on its own network.

This information is used in a variety of functions. Whenever any message goes out from any node on the network, all of the other stations recognize this and set the appropriate bit in the Active Station Table bit map table.

| **Word** | **Value** | **Meaning** |
|---|---|---|
| 23 | LO Byte | Active station table bit map, nodes 1 ... 8 |
| | HI Byte | Active station table bit map, nodes 9 ... 16 |
| 24 | LO Byte | Active station table bit map, nodes 17 ... 24 |
| | HI Byte | Active station table bit map, nodes 25 ... 32 |
| 25 | LO Byte | Active station table bit map, nodes 33 ... 40 |
| | HI Byte | Active station table bit map, nodes 41 ... 48 |
| 26 | LO Byte | Active station table bit map, nodes 49 ... 56 |
| | HI Byte | Active station table bit map, nodes 57 ... 64 |

The Token Station table bit map (Words 27 ... 30) is 64 bits that are dedicated in each station in the Modbus Plus Network. This is a table that identifies all of the active nodes on the network that are capable of receiving the token. The Active Station table bit map shows all of the nodes on the network versus the Token station table bit map shows only the nodes that are capable of receiving the token.

The appropriate bit in this table is set when the packets are sent and received. The bits are cleared when the station does not receive the token within a timeout period and a certain number of retries.

| **Word** | **Value** | **Meaning** |
|---|---|---|
| 27 | LO Byte | Token station table bit map, nodes 1 ... 8 |
| | HI Byte | Token station table bit map, nodes 9 ... 16 |
| 28 | LO Byte | Token station table bit map, nodes 17 ... 24 |
| | HI Byte | Token station table bit map, nodes 25 ... 32 |
| 29 | LO Byte | Token station table bit map, nodes 33 ... 40 |

| | | |
|---|---|---|
| | HI Byte | Token station table bit map, nodes 41 ... 48 |
| 30 | LO Byte | Token station table bit map, nodes 49 ... 56 |
| | HI Byte | Token station table bit map, nodes 57 ... 64 |

The Global Data present table bit map (Words 31 ... 34) signifies that there is Global Out Data available on each individual node. The PLC sets the appropriate bit when it is informed that there is Global Data available on a particular node. When the PLC is aware that there is Global Data available on a particular node, it can then request the data from that node. Unless the bit is set in this table the PLC can not request data from that node.

| **Word** | **Value** | **Meaning** |
|---|---|---|
| 31 | LO Byte | Global Data present table bit map, nodes 1 ... 8 |
| | HI Byte | Global Data present table bit map, nodes 9 ... 16 |
| 32 | LO Byte | Global Data present table bit map, nodes 17 ... 24 |
| | HI Byte | Global Data present table bit map, nodes 25 ... 32 |
| 33 | LO Byte | Global Data present table bit map, nodes 33 ... 40 |
| | HI Byte | Global Data present table bit map, nodes 41 ... 48 |
| 34 | LO Byte | Global Data present table bit map, nodes 49 ... 56 |
| | HI Byte | Global Data present table bit map, nodes 57 ... 64 |

The Receive Buffer in use bit map table (Words 35 ... 37) monitors the state of the receive buffers. Multiple receive buffers are available to process the input information that comes into the PLC. There is a need for more than one receive buffer because there is more than one path of data input to a PLC, and also because the PLC might need to leave the information in the buffer for a period of time.

When the in-use bits are set they designate that the buffer is occupied with data and is being used. Each data buffer is 288 bytes in size and there are 34 buffers.

| **Word** | **Value** | **Meaning** |
|---|---|---|
| 35 | LO Byte | Receive Buffer in use bit map, buffer 1 ... 8 |
| | HI Byte | Receive Buffer in use bit map, buffer 9 ... 16 |
| 36 | LO Byte | Receive Buffer in use bit map, buffer 17 ... 24 |
| | HI Byte | Receive Buffer in use bit map, buffer 25 ... 32 |
| 37 | LO Byte | Receive Buffer in use bit map, buffer 33 ... 34 |
| | HI Byte | Station management command processed initiation counter |

### G.1.21 Word 38 ... 53

Words 38 to 53 are counters for each individual path. There are Programming paths and Data paths.

Programming paths are used to transfer information from a programming panel (Like a PC running Modsoft or Concept panel software). When you are making programming changes, these changes are done over the Programming paths.

Data paths are the main workhorse. All of the data that needs to be transferred between nodes, whether it is a human interface package, a third party software package, or any other Modbus Plus compatible package, uses the data paths.

Each type of path (Programming or Data) may exist as either a Master or Slave path.

Master paths are paths that are the output from the particular device. For example, a Data Master path is data that is the output from the PLC and is being sent to other devices. An MSTR block that sends data out uses a Data Master Path.

Slave paths are paths that receive data from other devices. A Data Slave path is used for data that is being received by the device. For example a PLC is programmed via the Programming Slave Path. The PC used to do the programming attaches to the Programming Slave path and programs the PLC.

The following words are counters that show the number of transactions for all paths. Each time one message is sent or received in any of the paths, the path command count increases.

PLCs have four data master paths, four data slave paths, one programming master path, and one programming slave path.

SA85, PCMCIA cards, BP85 and the like have eight data master, eight data slave, eight programming master, and eight programming slave paths. These have more capabilities so that they can handle greater amounts of throughput.

| Word | Value | Meaning |
|---|---|---|
| 38 | LO Byte | Data master output path 1 command initiation counter |
|  | HI Byte | Data master output path 2 command initiation counter |
| 39 | LO Byte | Data master output path 3 command initiation counter |
|  | HI Byte | Data master output path 4 command initiation counter |
| 40 | LO Byte | Data master output path 5 command initiation counter |
|  | HI Byte | Data master output path 6 command initiation counter |
| 41 | LO Byte | Data master output path 7 command initiation counter |
|  | HI Byte | Data master output path 8 command initiation counter |

| Word | Value | Meaning |
|---|---|---|
| 42 | LO Byte | Data slave input path 41 command initiation counter |
|  | HI Byte | Data slave input path 42 command initiation counter |
| 43 | LO Byte | Data slave input path 43 command initiation counter |
|  | HI Byte | Data slave input path 44 command initiation counter |
| 44 | LO Byte | Data slave input path 45 command initiation counter |
|  | HI Byte | Data slave input path 46 command initiation counter |
| 45 | LO Byte | Data slave input path 47 command initiation counter |
|  | HI Byte | Data slave input path 48 command initiation counter |

| Word | Value | Meaning |
|---|---|---|
| 46 | LO Byte | Program Master output path 81 command initiation counter |
|  | HI Byte | Program Master output path 82 command initiation counter |
| 47 | LO Byte | Program Master output path 83 command initiation counter |
|  | HI Byte | Program Master output path 84 command initiation counter |

| Word | Value | Meaning |
|---|---|---|
| 48 | LO Byte | Program master output path 85 command initiation counter |
| | HI Byte | Program master output path 86 command initiation counter |
| 49 | LO Byte | Program master output path 87 command initiation counter |
| | HI Byte | Program master output path 88 command initiation counter |

| Word | Value | Meaning |
|---|---|---|
| 50 | LO Byte | Program Slave input path C1 command processed counter |
| | HI Byte | Program Slave input path C2 command processed counter |
| 51 | LO Byte | Program Slave input path C3 command processed counter |
| | HI Byte | Program Slave input path C4 command processed counter |
| 52 | LO Byte | Program Slave input path C5 command processed counter |
| | HI Byte | Program Slave input path C6 command processed counter |
| 53 | LO Byte | Program Slave input path C7 command processed counter |
| | HI Byte | Program Slave input path C8 command processed counter |

# Appendix H.
# Updating Your Product

# H.1 Updating Your Host Based Device

The internal operating code (the Executive program) for certain host based devices is resident in onboard memory chips that can be reloaded from an external source. This allows the device to be updated to newer versions as they become available. Updating is performed by downloading the new version to your PC from Modicon, and then loading the new version into your host based device.

This Appendix describes how to perform the update process.

## H.1.1 Products Affected

The information is only relevant to the following product:

□ AM-0984-A T4 Programmable Controller

## H.1.2 What You Need for Updating

**1** Before performing any download you should determine which version of the Executive program is currently resident in your device.

**2** Before performing the download you should determine the latest available version of the Executive program from Modicon.

**3** To download the new version, you will need to access the Modicon Customer Service Bulletin Board.

**4** To load the new Executive into your host based device, you will need a copy of the Modicon Executive Loader Utility program. You can obtain this by downloading it from the Modicon Customer Service Bulletin Board.

Procedures for performing the update are described on the following pages.

For additional technical assistance, you can call the Modicon Field Support Center at (800) 468-5342 (inside U.S. and Canada), or (508) 794-0800. You can also obtain information from your local Modicon representative.

### H.1.3 Determining Your Current Version

If you are using the Modicon MODSOFT software, you can see the current Executive program version on your CONTROLLER STATUS INFORMATION screen. The version is shown in the EXEC ID line on your screen.

For example: `EXEC ID 0861 REV 0101` identifies version 1.01.

Refer to your MODSOFT guidebook for further information.

If you are not using MODSOFT, you can access the version information through the use of Modbus commands in your application program. The version is stored in word 1 of the data returned by the Get Network Statistics command (Modbus function code 08, subfunction 21). This command is described in Appendix F.

### H.1.4 Determining the Latest Available Version

You can determine the latest available version of your Executive program by contacting the Modicon Field Support Center and accessing the Modfax service.

Modfax is an automatic document retrieval system for Modicon customers. The system is self-prompting. To access Modfax, call the Modicon Field Support Center at (800) 468-5342 (inside U.S. and Canada), or (508) 794-0800. Select option 3 when prompted. Have your FAX number ready.

### H.1.5 Downloading the New Version

You can download the latest available version of your Executive program by accessing the Modicon Customer Service Bulletin Board. The calling procedure is described on the next page.

### H.1.6 Downloading the Executive Loader Utility

You can download the Executive Loader Utility program by accessing the Modicon Customer Service Bulletin Board. The calling procedure is described on the next page.

## H.2 Accessing the Customer Service Board

The Customer Service Bulletin Board service operates continuously, 24 hours a day, without charge.

Here is the procedure for calling:

**1** Using your PC modem and telecommunication software package, dial: (508) 975-9779.

| Modem Parameter | Range |
|---|---|
| Baud rate | Up to 14400 |
| Parity mode | None |
| Data bits | 8 |
| Stop bits | 1 |

**2** If this is your first call to the Modicon Bulletin Board service, you will need to create an account. To do this, answer the questions you will be asked at this time.

**3** At your main menu, select the type of service you want. The menu is self-prompting.

If you are requesting a download of the Executive program, continue as follows:

**4** Your next menu shows a selection of Modicon products. Select the menu choice corresponding to the product you have.

**5** Your screen displays a list of available files, with a description of each file. Select the file with the latest version for your product.

**6** Select the download protocol that matches your telecommunication protocol -- for example ZMODEM, KERMIT, or XMODEM.

**7** Depending on your telecommunication software, you might have to enter some additional information for commencing the download. When your software is ready, the download will proceed.

**8** You should now have the downloaded file in your PC, in the path determined by your telecommunication software.

## H.3 Running the Executive Loader Utility

The Executive Loader Utility loads a binary Executive program file into your host based device. The utility contains five files:

□ LOADER.EXE -- the executable program that performs the loading function

□ LOADER.HLP -- the Help text file for the Loader utility

□ LOADER.NDX -- an index file for Help screens

□ MCMIII.MSG -- an error message file

□ README.1ST -- a text file that explains how to perform the update

Read the information in your README.1ST text file for instructions on performing the update. You might want to printout a hard copy of this file for reference.

# Index