BASEstar™ Open DAS for Modicon® on OpenVMS™

Installation and User's Guide

Order Number: AA-Q5FTD-TE

April 2000

This manual describes how to install and use the BASEstar Open Device Access Software for Modicon Programmable Controllers.

Revision/Update Information:	This is a revised document.
Operating System and Version:	OpenVMS/Alpha Version 6.1
Operating System and Version:	OpenVMS/VAX Version 6.1
Interface Software and Version:	BASEstar Open Version 3.1
Software Version:	Device Access Software for Modicon Programmable Controllers, Version 3.1A

Compaq Computer Corporation
Houston, Texas

April 2000

© 1994 Compaq Computer Corporation.

COMPAQ, VAX and VMS Registered in U.S. Patent and Trademark Office.

BASEstar and OpenVMS are trademarks of Compaq Information Technologies Group, L.P. in the United States and/or other countries.

Modicon and Modbus are registered trademarks of Schneider Automation, Inc.

SQ85, SA85, Modbus Plus and Quantum Automation Series are trademarks of Schneider Automation, Inc.

All other products mentioned herein may be trademarks or registered trademarks of their respective companies.

Confidential computer software. Valid license from Compaq or authorized sublicensor required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentaion, and Technical data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Compaq shall not be liable for technical or editorial errors or omissions contained herein. The information in this document is subject to change without notice.

This document is available on CD–ROM.

This document was prepared using DECdocument, Version 3.3.

Contents

Preface vii

1 Overview

1.1 Description 1–1
1.2 Device Communications 1–1
1.3 Supported Functions and Devices 1–2

2 Installing and Configuring the DAS

2.1 Installation Requirements 2–1
2.1.1 Hardware 2–1
2.1.2 Software 2–4
2.1.3 Additional Software 2–4
2.1.4 Disk Space 2–5
2.2 Installing the DAS 2–5
2.2.1 Files Created During Installation 2–7
2.2.2 Installation Messages 2–7
2.3 Postinstallation Tasks 2–9
2.3.1 Editing the Configuration Files 2–9
2.3.1.1 Editing ODS Registrations 2–10
2.3.1.2 Editing BASEstar Open Protocol Profile Definitions 2–13
2.3.1.3 Editing VMD Definitions 2–13
2.3.1.4 DAS Configuration Files 2–15
2.3.2 Configuring Ports 2–18
2.3.3 Configuring TELNET Access 2–18
2.3.4 Setting Up Plant-Floor Equipment 2–19
2.3.5 Tracing Device Communications 2–19
2.4 Failures During Product Use 2–20

3 Using the DAS

3.1 Accessing Modicon Functions 3–1
3.2 Supported Functions 3–1
3.2.1 Connect, Conclude and Abort Functions 3–2
3.2.2 Read and Write Variable Functions 3–2
3.2.3 Read and Write Data Point Functions 3–5
3.2.4 Receiving Indications 3–7
3.2.4.1 Write Indications 3–7
3.2.4.2 Abort Indications 3–11
3.3 Addressing Syntax and Supported Data Types 3–11
3.3.1 Modicon Micro 84 Addressing Syntax and Supported Data Types 3–11
3.3.2 Modicon 184 and 384 Addressing Syntax and Supported Data Types 3–12

3.3.3 Modicon 484 Addressing Syntax and Supported Data Types 3–12
3.3.4 Modicon 884 Addressing Syntax and Supported Data Types 3–12
3.3.5 Modicon 584 and 984 Addressing Syntax and Supported Data Types 3–13

A I/O Error Messages

B DAS Error Messages

Index

Examples

2–1 ODS Registration 2–10
2–2 BASEstar Open Protocol Profile Configuration 2–13
2–3 VMD Configuration 2–13
2–4 ODS Configuration File 2–15
2–5 BSTR Configuration File 2–16
3–1 UNNAMED_VARIABLE CREATE Command 3–3
3–2 UNNAMED_VARIABLE GET VALUE Output 3–4
3–3 UNNAMED_VARIABLE PUT VALUE Output 3–5
3–4 DEVICE_DATA_POINT CREATE Command 3–5
3–5 DATA_POINT GET VALUE Output 3–6
3–6 DATA_POINT PUT VALUE Output 3–7
3–7 Modbus Plus Write Indication Configuration 3–8
3–8 Modbus TCP/IP Write Indication Configuration 3–9
3–9 Modbus Write Indication Configuration 3–9
3–10 MOD.PAR Contents 3–9
3–11 Write Indication Example 3–10

Figures

1–1 DAS Communications 1–2
2–1 Modbus Communications 2–2
2–2 Modbus RS-232 Pinouts 2–2
2–3 Modbus Plus Communications 2–3
2–4 Modbus TCP/IP Communications 2–4

Tables

2–1 Disk Space Requirements 2–5
2–2 Files Created During Intallation 2–7
2–3 Modbus RS-232 REGISTER Attributes 2–10
2–4 Modbus Plus REGISTER Attributes 2–11
2–5 TCP/IP REGISTER Attributes 2–12
2–6 VMD CREATE 2–14
3–1 Connect, Conclude and Abort Interface Access 3–2

3–2	Read and Write Variable Types	3–2
3–3	Read and Write Variable Interface Access	3–3
3–4	Read and Write Data Point Interface Access	3–5
3–5	Indication Interface Access	3–7
3–6	Modicon Micro 84 Address Format	3–11
3–7	Modicon 184 and 384 Address Format	3–12
3–8	Modicon 484 Address Format	3–12
3–9	Modicon 884 Address Format	3–12
3–10	Modicon 584 and 984 Address Format	3–13

Preface

This document describes how to install and use the BASEstar Open DAS for Modicon Programmable Controllers.

Intended Audience

This document is intended for system managers who must set up and maintain the following:

- BASEstar Open software
- BASEstar Open DAS for Modicon Programmable Controllers

This document is also intended for application programmers who develop plant-floor management software layered on BASEstar Open software.

Readers of this document should have knowledge of:

- OpenVMS operations and administration
- OpenVMS application software
- BASEstar Open software
- Site-specific installation requirements

In addition, knowledge of the Modicon programmable controllers is required.

Document Structure

This document is organized as follows:

- Chapter 1 provides an overview of the DAS for Modicon programmable controllers.
- Chapter 2 provides the information you need to install and configure the DAS for Modicon programmable controllers.
- Chapter 3 provides information about the supported functions for Modicon programmable controllers, and how to access the functions.

Associated Documents

For information on installing and configuring a BASEstar Open system, refer to the following documents:

- *BASEstar Open for OpenVMS Release Notes*
- *BASEstar Open for OpenVMS Installation Guide*
- *BASEstar Open for OpenVMS Management Guide*

For information on installing and configuring a BASEstar Open system, refer to the following documents:

- *BASEstar Open for OpenVMS Release Notes*
- *BASEstar Open for OpenVMS Installation Guide*
- *BASEstar Open for OpenVMS Management Guide*

For information on using BASEstar Open software functions, refer to the following documents:

- *BASEstar Open for OpenVMS Introduction*
- *BASEstar Open for OpenVMS Reference Guide (Model)*
- *BASEstar Open for OpenVMS Command Language Interface*
- *BASEstar Open for OpenVMS Error Messages Guide*

For more information on developing software to integrate manufacturing applications and equipment, refer to the following documents:

- *BASEstar Open for OpenVMS Application Programming Interface*

For information on installing and using OMNI functions, refer to the following documents:

- *Compaq OMNI for OpenVMS Installation Guide*
- *Compaq OMNI for OpenVMS Application Programmer's Interface*
- *Compaq OMNI for OpenVMS Guide to Using Omni Directory Services*
- *Compaq OMNI for OpenVMS Guide to Using OmniView*

For information on Modicon programmable controller communications and addressing, refer to the following documents:

- *Modicon Protocol Reference Guide*
- *Modicon DEC Host Based Devices Users Guide*

Conventions

This document uses the following conventions:

Boldface	Highlights user input within textual descriptions.
Return	Press the key labeled Return. Unless otherwise specified, press Return after entering a command or responding to a prompt.
Enter	Type the words or symbols described and press Return.

1
Overview

This chapter provides an overview of the BASEstar Open DAS for Modicon Programmable Controllers. It also briefly describes Modicon programmable controller communications, and the supported functions for the DAS for Modicon programmable controllers.

1.1 Description

The DAS for Modicon programmable controllers allows you to access Modicon programmable controllers using BASEstar Open Device Services, BASEstar Open Data Services or OMNI Services.

Device Services is the device configuration and data collection component of BASEstar Open for OpenVMS, allowing the user to:

- Configure device and variable definitions
- Read and write variable values
- Configure polled data collection

Data Services is the data management component of BASEstar Open for OpenVMS that provides device independent access to shop floor data allowing the user to:

- Configure data points
- Read and write data point values
- Generate events based on data changes

OMNI Services provides the Applications Programming Interface (API) used by the BASEstar Open Device/Data Services to communicate with shop floor devices. OMNI uses the Manufacturing Messaging Specification (MMS) (ISO-9506) model to communicate with shop floor devices. OMNI also has the capability to support non-MMS devices such as Modicon programmable controllers through protocol specific Device Access Software, such as the DAS for Modicon programmable controllers.

Using the DAS for Modicon programmable controllers, applications can perform a variety of device access functions for the Modicon programmable controllers.

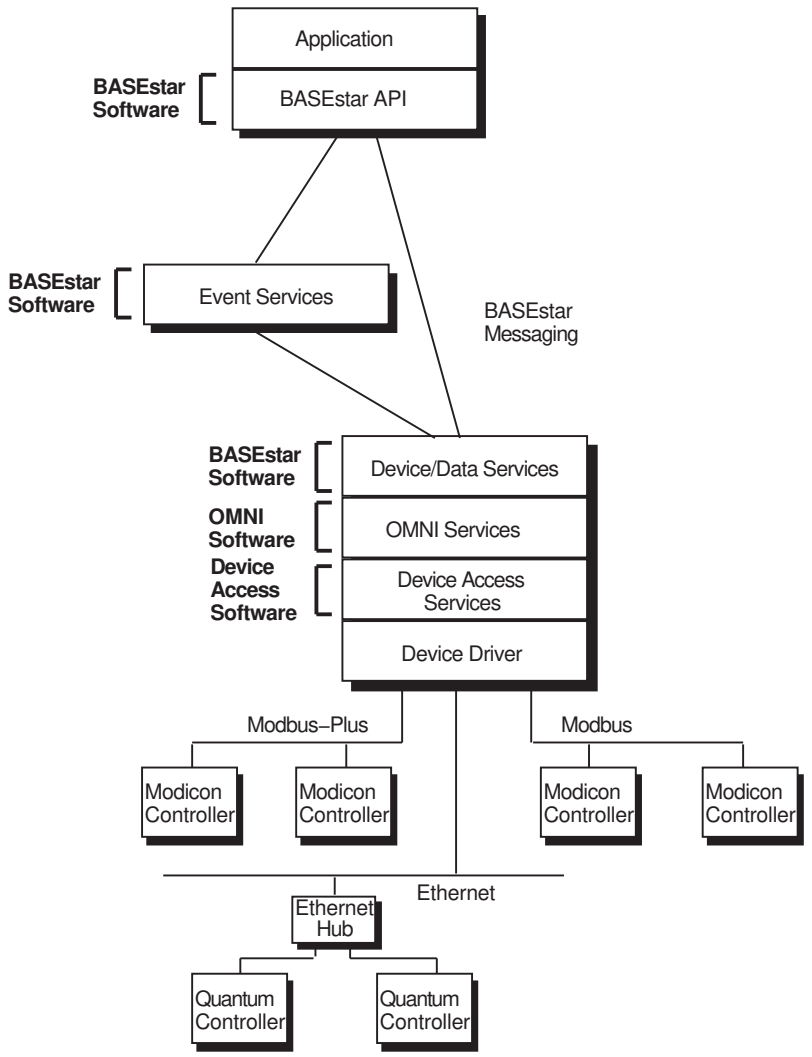
1.2 Device Communications

The DAS for Modicon programmable controllers consists of software that provides device-specific communications for users of the OMNI API. BASEstar Open Device/Data Services use the OMNI API to provide device communications to BASEstar Open users.

1.2 Device Communications

The DAS software communicates the requests and data directly with an OpenVMS device driver to send data to and receive data from plant-floor devices. Figure 1–1 shows how the DAS for Modicon programmable controllers facilitates communications between BASEstar Open Device or Data Services and the device.

Figure 1–1 DAS Communications



1.3 Supported Functions and Devices

You can perform only device access funtions that are supported by the DAS.

The DAS for Modicon programmable controllers supports the following OMNI Services functions for Modicon programmable controllers:

- Initiate a connection to a device (omni_connect)
- Conclude a connection with a device (omni_conclude)
- Abort a connection with a device (omni_abort)
- Get a value from a device's memory (omni_get_value)

- Put a value to a device's memory (omni_put_value)
- Receive a write indication from a device (omni_get_indications)
- Receive an abort from a device (omni_get_indications)

The DAS for Modicon programmable controllers supports the following Modicon programmable controllers:

- Modicon Micro 84 controllers
- Modicon 184 controllers
- Modicon 384 controllers
- Modicon 484 controllers
- Modicon 584 controllers
- Modicon 884 controllers
- Modicon 984 controllers
- Modicon Quantum Automation Series controllers (when configured as Modicon controllers)
- Modicon Q984 (Modbus Plus) (VAX)

For more information about the supported functions for these programmable controllers, refer to Chapter 3 of this document.

2
Installing and Configuring the DAS

This chapter provides the information you need to install the DAS for Modicon programmable controllers and to configure your system.

2.1 Installation Requirements

Review the following hardware and software requirements to ensure that your system is prepared for the DAS for Modicon programmable controllers installation.

2.1.1 Hardware

The hardware requirements for the DAS for Modicon programmable controllers are the same as those for the BASEstar Open software. For specific hardware requirements, refer to the *BASEstar Open for OpenVMS Installation Guide* .

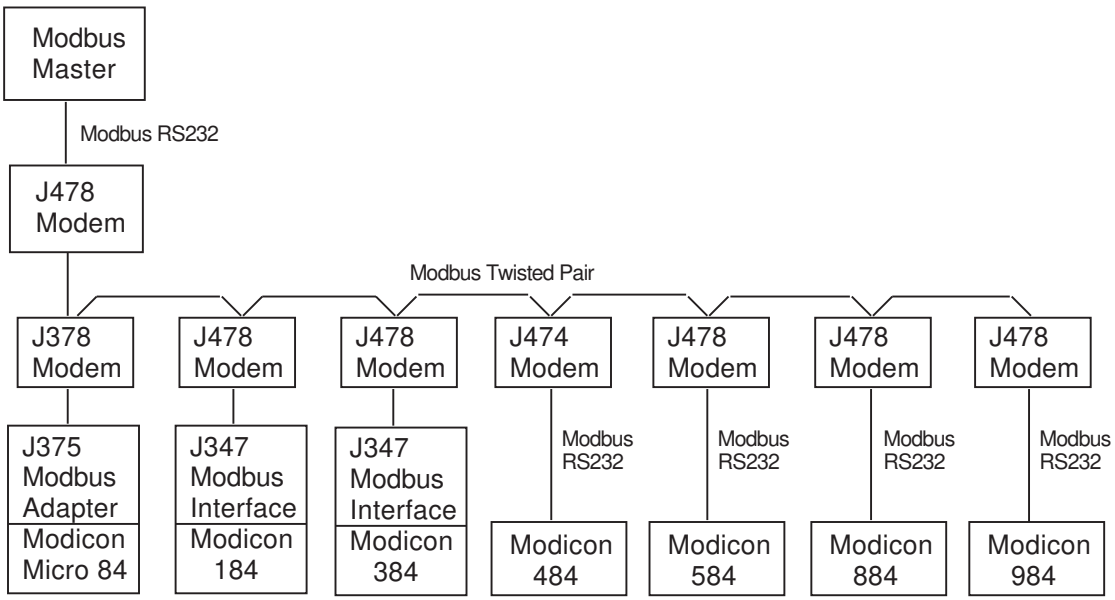
Additional hardware required for configuring the DAS for Modicon programmable controllers using Modbus includes:

- Any supported Modicon programmable controller.
- J478 Modbus modem to connect to the OpenVMS system.
- J478 Modbus modem to connect to the programmable controller.
- A cable from the OpenVMS system to the Modbus modem.
- J374 Modbus communications adapter if using Modicon 184 or 384 programmable controllers.
- J474 Modbus communications adapter if using Modicon 484 programmable controllers.

Installing and Configuring the DAS
2.1 Installation Requirements

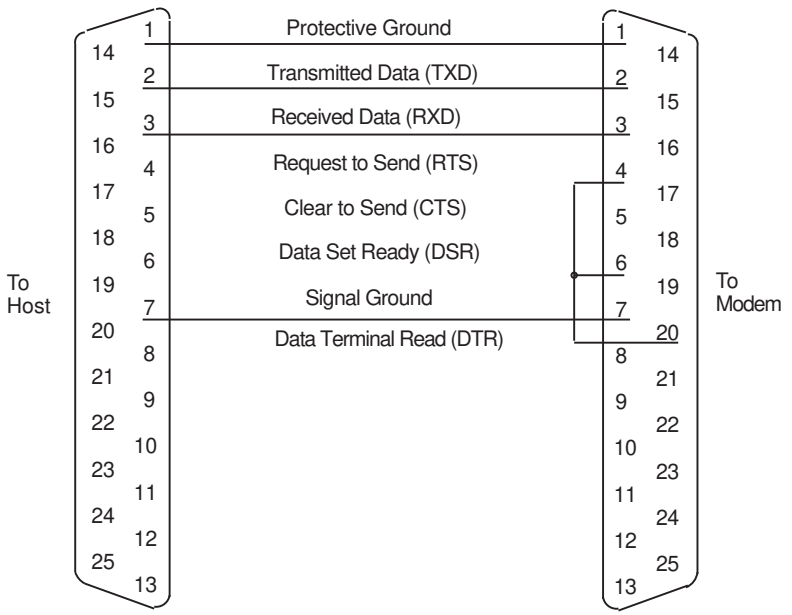
Figure 2–1 shows the required hardware for each model of programmable controller.

Figure 2–1 Modbus Communications



The cable to the J478 Modbus modem must be terminated with a 25 pin male RS232-C connector with pinouts as shown in Figure 2–2.

Figure 2–2 Modbus RS-232 Pinouts



Additional hardware required for configuring the DAS for Modicon programmable controllers using Modbus Plus includes:

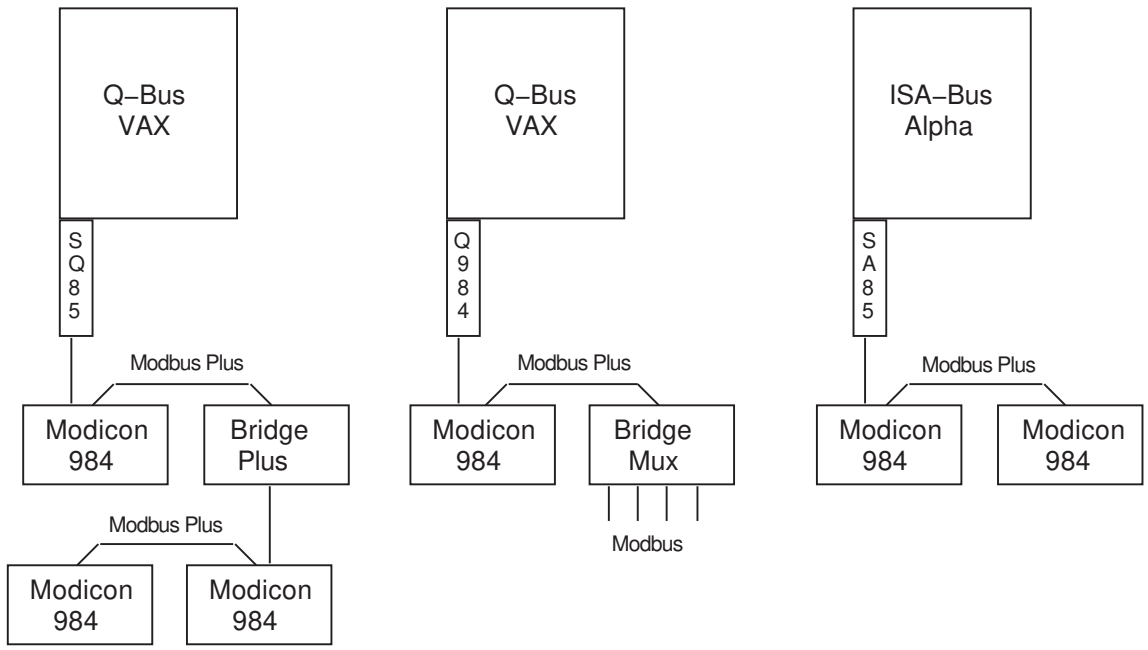
- Any supported Modicon 984 programmable controller model that ends in the number 5 (for example Modicon 984/385)
- Modicon SQ85 Modbus Plus Network Adapter or Q984 programmable controller (VAX)
- Modicon SA85 Modbus Plus Network Adapter (Alpha)

The SQ85 or Q984 card fits only into a VAX Q-BUS backplane.

The SA85 card fits only into an Alpha ISA backplane.

Figure 2–3 shows the hardware required to use Modbus Plus.

Figure 2–3 Modbus Plus Communications



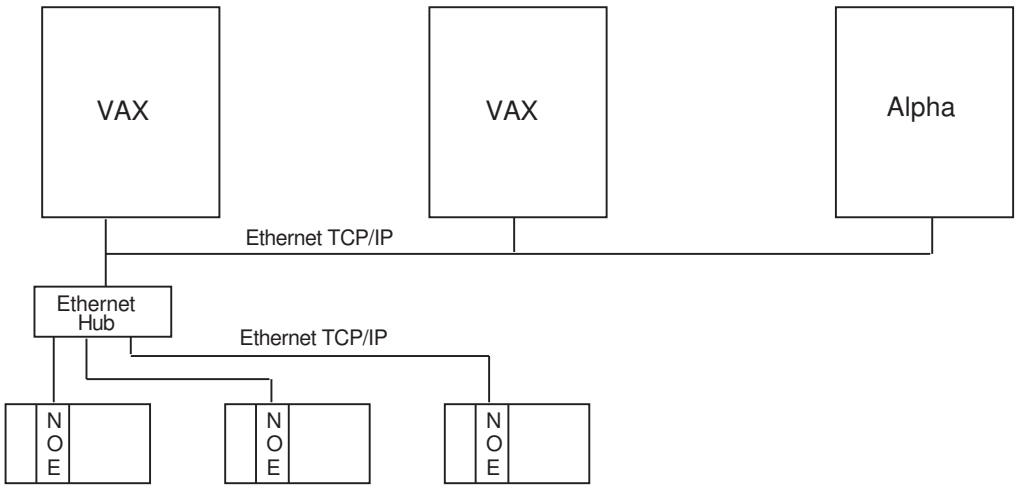
Additional hardware required for configuring the DAS for Modicon programmable controllers using Modbus over TCP/IP includes:

- Any supported Modicon Quantum Automation Series programmable controller model that supports the Modicon Quantum Ethernet TCP/IP Module
- Modicon Quantum Ethernet model 140 NOE 211 00 (twisted pair) module
- Modicon Quantum Ethernet model 140 NOE 251 00 (fiber optic) module
- Ethernet hub

Quantum Automation Series programmable controllers are supported by the DAS using the Modicon 984 protocol profile(s).

Figure 2–4 shows the hardware required to use Modbus over TCP/IP.

Figure 2–4 Modbus TCP/IP Communications



2.1.2 Software

The following software must be installed prior to installing DAS for Modicon programmable controllers:

- OpenVMS/Alpha Version 6.1
- OpenVMS/VAX Version 6.1
- BASEstar Open for OpenVMS Version 3.1

For more information on installing BASEstar Open software, refer to the *BASEstar Open for OpenVMS Installation Guide*.

Note

Before using this product on a system, you must first register a License Product Authorization Key (License PAK) using the License Management Facility (LMF). For more information about the License Management Utility, refer to the *License Management Utility Manual* for OpenVMS.

2.1.3 Additional Software

In addition to the above software, if Modbus Plus is being used, the following software must also be installed:

- Modbus Plus driver for OpenVMS/VAX (SQ85 network card)
- Modbus Plus driver for OpenVMS/Alpha (SA85 network card)

The Modbus Plus driver for OpenVMS/VAX is obtained from Schneider Automation, Inc. The Modbus Plus driver for OpenVMS/Alpha is obtained from IPACT (Integrated Process Automation and Control Technologies).

In addition to the above software, if Modbus TCP/IP is being used, the following software must also be installed:

- TCP/IP Services for OpenVMS

2.1.4 Disk Space

Table 2–1 lists the disk space required to install the DAS for Modicon programmable controllers. The space requirements are approximations; actual sizes may vary depending on your system environment and configuration.

Table 2–1 Disk Space Requirements

Usage	Approximate Space Requirements (Blocks)
Peak (during installation)	950 (Alpha)
	3000 (VAX)
Net (after installation)	450 (Alpha)
	1500 (VAX)

2.2 Installing the DAS

When your system meets all hardware and software requirements, you can install the DAS for Modicon programmable controllers. The installation takes from 1 to 5 minutes, depending on your system load and configuration.

Install the DAS for Modicon programmable controllers by using the following steps:

1. Log in to a privileged system manager's account.
2. Set the default directory to SYS$UPDATE:

   ```
   $  SET DEFAULT SYS$UPDATE
   ```

3. Invoke VMSINSTAL:

   ```
   $  @SYS$UPDATE:VMSINSTAL MODVVA031 ddcu:
   ```

   where:

   - **MODVVA** argument is the kit name
   - **VV** in the name means this is an OpenVMS/VAX kit
   - **VA** in the name means this is an OpenVMS/Alpha kit
   - **031** portion of the name is the version number
   - **ddcu** argument represents the name of the device on which the installation media is mounted, where:
     - **dd** is the device code
     - **c** is the controller designation
     - **u** is the unit number

VMSINSTAL prompts you for information during the installation. Note that DECnet software does not need to be running to perform the installation procedure. The following is an example of the output from the installation:

```
        OpenVMS VAX Software Product Installation Procedure V7.2

It is 22-MAR-2000 at 09:59.

Enter a question mark (?) at any time for help.

* Are you satisfied with the backup of your system disk [YES]?
```

```
The following products will be processed:

  MODVVA V3.1

       Beginning installation of MODVVA V3.1 at 10:00

%VMSINSTAL-I-RESTORE, Restoring product save set A ...
%VMSINSTAL-I-RELMOVED, Product's release notes have been moved to SYS$HELP.


   Copyright 1994 Compaq Computer Corporation

   Confidential computer software.  Valid license from Compaq or authorized
   sublicensor required for possession, use or copying.  Consistent with FAR
   12.211 and 12.212, Commercial Computer Software, Computer Software
   Documentation, and Technical Data for Commercial Items are licensed to the
   U.S. Government under vendor's standard commercial license.

   BASEstar Open DAS for Modicon Programmable Controllers
   installation procedure.

   Checking for a valid license...


        Product:      MODDAS-V
        Producer:     DEC
        Version:      1.0
        Release Date: 01-JUN-1994

* Does this product have an authorization key registered and loaded? y

   Now checking OpenVMS version...

   Now checking that OMNI is installed...

   Now checking OMNI version...

   Now checking OMNI logicals...

   Now checking the OMNI integrator registration utility...

* Do you want BASEstar Open support [YES]?

   Now checking disk space...

* Do you want to purge files replaced by this installation [YES]?
* Do you want to run the IVP after the installation [YES]?

   ---------------------------------NOTE-----------------------------------

   During installation the file ODS_MOD_CONFIG.TEMPLATE is copied to the
   ODS:[SCRIPTS] directory.  Copy and edit this file to create site specific
   /PATH and /LINE directory definitions for devices at your installation.

   ---------------------------------NOTE-----------------------------------

   During installation the file BSTR_MOD_CONFIG.TEMPLATE is copied to the
   BSTR$KIT:[ETC] directory.  Copy and edit this file to create site specific
   VMD definitions.

   Now modifying the ODS schema to add Modicon specific
   object classes and attributes...

Checking ODS Attribute Type Schema definitions ...

Checking ODS Object Class Schema definitions ...

Querying ODS Schema definitions ...
```

```
Valid Schema definitions found in the attribute schema database as:
OC ALIASED_NAME CN C L S O OU P_ADDR APC OWNER APT AEQ MAPV SY OBJ_ID BER PSAP S
SAP TSAP N_ADDR AS_LIST DESCR VMD DOM DEL VN APPSN APROF VERSION NLEVEL MAX_CALL
ED MAX_CALLING MAX_SEG CBB_LIST SS_LIST VENDOR MODEL REV INUM DN SHARE CONT_FILE
 CAP_FILE PN REUSE ARG MON DOM_LIST VAR ATYPE ADDR_T ADDR_V PATH LINE STATION LI
NE_NAME PORT ENQ_RETRIES NAK_RETRIES TIMEOUT SPEED PARITY MODEM TRACE KE_STATION
 KE_TIMEOUT SLAVE_ADDRESS RETRIES SLAVE_ROUTING NODE_ADDRESS

Valid Schema definitions found in the object class schema database as:
TOP ALIAS C L O OU AP AE DSA DEVICE TAE OAE AS VMD DOMAIN PI NVAR UNVAR DH_LINE
DH_PATH MODBUS_PATH MODPLUS_SLAVE_PATH MODPLUS_NODE_PATH MODBUS_LINE MODPLUS_LIN
E

Normal Completion.
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...

   Now registering BASEstar Open DAS for Modicon Programmable Controllers
   with OMNI...

   Copyright 1994 Compaq Computer Corporation

   Executing the Installation Verification Procedure.

   BASEstar Open DAS for Modicon Programmable Controllers installation
   procedure has succeeded.

        Installation of MODVVA V3.1 completed at 10:01

        VMSINSTAL procedure done at 10:01
```

2.2.1 Files Created During Installation

Table 2–2 lists the files created by the DAS for Modicon programmable controllers installation procedure, and the directories in which those files are placed.

Table 2–2 Files Created During Intallation

Directory	Filename
ODS:[SCRIPTS]	ODS_MOD_CONFIG.TEMPLATE
BSTR$KIT:[ETC]	BSTR_MOD_CONFIG.TEMPLATE

2.2.2 Installation Messages

You may see VMSINSTAL messages during the installation procedure. The following messages are specific to the DAS for Modicon programmable controllers installation:

BADOMNI, The version of OMNI required is OMNI V3.1 or later.,

Explanation: Error. This version of the DAS is designed to work with OMNI V3.1 or later.

User Action: Install OMNI V3.1 or later and then reinstall the DAS.

BADVMS (Alpha), BASEstar Open DAS for Modicon Programmable Controllers must be installed under OpenVMS V6.1.,

Explanation: Error. The DAS is being installed on a version of OpenVMS that is less than OpenVMS V6.1.

User Action: Upgrade OpenVMS to the required minimum version and then install the DAS.

BADVMS (VAX), BASEstar Open DAS for Modicon Programmable Controllers must be installed under OpenVMS V6.1.,

Explanation: Error. The DAS is being installed on a version of OpenVMS that is less than OpenVMS V6.1.

User Action: Upgrade OpenVMS to the required minimum version and then install the DAS.

NETBLOCKS (VAX), BASEstar Open DAS for Modicon Programmable Controllers requires 1500 blocks after installation.,

Explanation: Error. There isn't enough space on the system disk to install the DAS.

User Action: Delete unneeded files and reattempt the installation.

NETBLOCKS (ALPHA), BASEstar Open DAS for Modicon Programmable Controllers requires 450 blocks after installation.,

Explanation: Error. There isn't enough space on the system disk to install the DAS.

User Action: Delete unneeded files and reattempt the installation.

NOETC, Unable to find BSTR$KIT:[ETC] directory.,

Explanation: Error. The directory to place the BASEstar Open configuration template for Modicon devices was not found.

User Action: Reinstall BASEstar Open to set up the BSTR$KIT:[ETC] directory.

NOLICENSE, No license found for this product - IVP will not be run.,

Explanation: Informational. A valid license was not found. The installation will continue, but the IVP will not be run.

User Action: Register and load a valid license for this product before attempting to use the DAS.

NOLOAD, License for this product not loaded - IVP will not be run.,

Explanation: Informational. The license for this product has not been loaded by the License Management Utility. The installation willl proceed, but the IVP will not be run.

User Action: Load the license using the License Management Utility before attempting to use the DAS.

NOOMNI, OMNI must be installed before BASEstar Open DAS for Modicon Programmable Controllers.,

Explanation: Error. The installation procedure could not verify that DEComni was installed on the system. OMNI installation is a prerequisite to the installation of this DAS.

User Action: Install OMNI V3.1 and then reinstall the DAS.

NOINTREG, Unable to find the integrator registration utility.,

Explanation: Error. OMNI provides a registration utility for DASes to use to register their presence with DEComni. This registration utility could not be found.

User Action: Reinstall OMNI to provide the required registration utility.

NOODS, Unable to find the directory for OMNI Directory Services.,

Explanation: Error. The installation procedure creates classes and attributes in the OMNI Directory Services database and this database could not be found.

User Action: Reinstall OMNI to set up the OMNI Directory Services directory.

NOSCRIPT, Unable to find ODS:[SCRIPTS] directory.,

Explanation: Error. The directory to place the ODS configuration template for Modicon devices was not found.

User Action: Reinstall OMNI to set up the OMNI Directory Services directory.

ODSINI, OMNI logicals have not been defined, attempting to define logicals.,

Explanation: Information. The DAS installation requires that certain OMNI logicals be defined and these logical names were not found. The installation procedure attempts to define the logical names by executing SYS$STARTUP:OMNI_LOGICALS.COM.

User Action: None.

2.3 Postinstallation Tasks

Before using the DAS for Modicon programmable controllers you must perform the following postinstallation tasks:

- Edit the configuration files
- Configure ports
- Set up plant-floor equipment

2.3.1 Editing the Configuration Files

Configuration files, ODS_MOD_CONFIG.TEMPLATE and BSTR_MOD_CONFIG.TEMPLATE, are supplied with the DAS for Modicon programmable controllers. The ODS configuration file contains OMNI Directory Services (ODS) registration commands for Modicon specific classes and attributes. The BASEstar Open configuration file contains definitions for sample Modicon VMDs and protocol profile(s).

Copy the template files and edit them to include site-specific directory, VMD and protocol profile definitions. When you complete your edits, execute the files to create the ODS and BASEstar Open Device Services definitions.

The following sections provide examples of ODS and BASEstar Open definitions from the configuration files, and indicate which attributes can be modified. For more information about creating VMD and protocol profile definitions, refer to the *BASEstar Open for OpenVMS Command Language Interface*. For more information about registering ODS definitions, refer to the *Compaq OMNI for OpenVMS Guide to Using Omni Directory Services*.

2.3.1.1 Editing ODS Registrations

ODS registrations allow the DAS software to locate a particular device on an Modicon Modbus network. Example 2–1 shows the configuration file commands that create the ODS registrations.

Example 2–1 ODS Registration

```
REGISTER DIRECTORY NAME "/LINE=MODBUS_LINE" ATTRIBUTES "OC=MODBUS_LINE
/PORT=LTA207:/RETRIES=3/SPEED=9600/PARITY=none/MODEM=0/TIMEOUT=3/TRACE=0"
REGISTER DIRECTORY NAME "/LINE=MODPLUS_LINE" ATTRIBUTES "OC=MODPLUS_LINE
/PORT=MPA1:/RETRIES=2/TIMEOUT=2/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MOD984_PATH" ATTRIBUTES "OC=MODBUS_PATH
/LINE_NAME=(/LINE=MODBUS)/SLAVE_ADDRESS=20/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MODPLUS984_SLAVE_PATH"
ATTRIBUTES "OC=MODPLUS_SLAVE_PATH/LINE_NAME=(/LINE=MODPLUS_LINE)
/SLAVE_ROUTING=03.00.00.00.00/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MODPLUS984_NODE_PATH"
ATTRIBUTES "OC=MODPLUS_NODE_PATH/LINE_NAME=(/LINE=MODPLUS_LINE)
/NODE_ADDRESS=4/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MOD884_PATH" ATTRIBUTES "OC=MODBUS_PATH
/LINE_NAME=(/LINE=MODBUS)/SLAVE_ADDRESS=21/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MOD584_PATH" ATTRIBUTES "OC=MODBUS_PATH
/LINE_NAME=(/LINE=MODBUS)/SLAVE_ADDRESS=22/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MOD484_PATH" ATTRIBUTES "OC=MODBUS_PATH
/LINE_NAME=(/LINE=MODBUS)/SLAVE_ADDRESS=23/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MODTCP984_PATH" ATTRIBUTES "OC=MODTCP_PATH
/IP_ADDRESS=19.20.20.5"
```

You can modify the commands for your site configuration. The following tables show ODS classes and attributes supportes by the DAS. Attribute values are not checked until a connection is initiated with the VMD.

Table 2–3 lists the classes and attributes that are supported for Modicon devices using Modbus over an RS-232 connection.

Table 2–3 Modbus RS-232 REGISTER Attributes

Class	Attribute	Default	Value(s)
MODBUS_LINE	PORT		TT01:, LTA201:, etc.[1]
	SPEED		110 - 9600
	PARITY		NONE, EVEN, ODD
	MODEM	NO	TRUE, ON, YES, 1 or FALSE, OFF, NO, 0
	TIMEOUT	3	2-10[2]
	RETRIES	3	0-20
	TRACE[3]	NO	TRUE, ON, YES, 1 or FALSE, OFF, NO, 0

[1]LTAXXX: is used for LAT connections. TTXX: is used for direct connection.

[2]Seconds.

[3]Trace is used to allow troubleshooting of the device communications. See Section 2.3.5 for details.

(continued on next page)

Table 2–3 (Cont.) Modbus RS-232 REGISTER Attributes

Class	Attribute	Default	Value(s)
MODBUS_TELNET	IP_ADDRESS		xx.xx.xx.xx[4]
	PORT_NUMBER[5]		
	TIMEOUT	3	2-10[2]
	RETRIES	3	0-20
	TRACE[3]	NO	TRUE, ON, YES, 1 or FALSE, OFF, NO, 0
MODBUS_PATH	LINE_NAME		/LINE=xxx[6]
	SLAVE_ADDRESS		1-247
	TRACE[3]	NO	TRUE, ON, YES, 1 or FALSE, OFF, NO, 0

[2]Seconds.

[3]Trace is used to allow troubleshooting of the device communications. See Section 2.3.5 for details.

[4]The TCP/IP address must be in xx.xx.xx.xx notation where xx is a number from 0-255.

[5]Put a valid port number for a socket connection to the TELNET server.

[6]Specify a name registered earlier with ODS that specifies the line that the programmable controller with this slave address will be using.

Table 2–4 lists the classes and attributes that are supported for Modicon devices using Modbus over a Modbus Plus connection.

Table 2–4 Modbus Plus REGISTER Attributes

Class	Attribute	Default	Value(s)
MODPLUS_LINE	PORT		MPA1:, MPB1:, MPC1:, MPD1:, JPA1:, JPB1:, JPC1:, JPD1:[1]
	RETRIES	3	0-20
	TIMEOUT	3	2-10[2]
	SLAVE_PATHS	1	0-8[3]
	MASTER_PATHS	7	1-8[4]
	IND_ROUTE_PATH[5]	NO	TRUE, ON, YES, 1 or FALSE, OFF, NO, 0
	TRACE[6]	NO	TRUE, ON, YES, 1 or FALSE, OFF, NO, 0

[1]Specify the OpenVMS Modbus Plus device name. A maximum of four cards are allowed per system. Use the MP or JP device name depending on the driver installed on your OpenVMS system.

[2]Seconds.

[3]Specify the number of Data Slave paths for the DAS to allocate. Slave paths are used to received MSTR function block requests from the device.

[4]Specify the number of Data Master paths for the DAS to allocate. Master paths are used to read and write data to the device.

[5]If set, this flag tells the DAS that the first five words of data in the message contain the routing path of the programmable controller that sent the message. (The five bytes of the routing path are in the lower 8 bits of the 16 bit words.) The DAS uses this information to locate the local VMD of the device that sent the message and uses that VMD to look up a variable into which to place the data. If the flag is not set, then the DAS assumes that the local VMD is the same for all VMDs on the line and uses the first local VMD on the line that it finds to look up the variable. With the current implementation of local VMDs in BASEstar Open, this flag should never be set.

[6]Trace is used to allow troubleshooting of the device communications. See Section 2.3.5 for details.

(continued on next page)

Table 2–4 (Cont.) Modbus Plus REGISTER Attributes

Class	Attribute	Default	Value(s)
MODPLUS_SLAVE_PATH	LINE_NAME		/LINE=xxx[7]
	SLAVE_ROUTING		xx.xx.xx.xx.xx[8]
	TRACE[6]	NO	TRUE, ON, YES, 1 or FALSE, OFF, NO, 0
MODPLUS_NODE_PATH	LINE_NAME		/LINE=xxx[9]
	NODE_ADDRESS		1-64[10]
	TRACE[6]	NO	TRUE, ON, YES, 1 or FALSE, OFF, NO, 0

[6]Trace is used to allow troubleshooting of the device communications. See Section 2.3.5 for details.

[7]Specify a name registered earlier with ODS that specifies the Modbus Plus port that the programmable controller with this slave routing will be using.

[8]The slave routing can contain up to 5 routing bytes. If using a bridge the nodes can be 1-64 where the address is node.node.node.node.node (if using the maximum number of bridges). If no bridge is used, then the address would look like node.0.0.0.0. All five routing bytes must be specified even if they are zero. If using a bridge multiplexor, then the last three bytes of the routing information looks like node.port.address where node is the address of the bridge multiplexor (1-64), port is the port on the multiplexor (1-4) and address is the Modbus address (1-247) of the programmable controller.

[9]Specify a name registered earlier with ODS that specifies the Modbus Plus port that is associated with this VMS Modbus Plus device.

[10]Specify the Modbus Plus node address for the OpenVMS node.

Table 2–5 lists the classes and attributes that are supported for Modicon devices using Modbus over a TCP/IP connection.

Table 2–5 TCP/IP REGISTER Attributes

Class	Attribute	Default	Value(s)
MODTCP_PATH	IP_ADDRESS		xx.xx.xx.xx[1]
	PROBE_TIME[2]	10	
	DROP_TIME[3]	30	
	TIMEOUT	3	1-10[4]
	TRACE[5]	NO	TRUE, ON, YES, 1 or FALSE, OFF, NO, 0

[1]The TCP/IP address must be in xx.xx.xx.xx notation where xx is a number from 0-255. The port number is fixed, so does not need to be specified.

[2]Probe time is the time (in seconds) that TCP/IP will use to probe for a broken connection.

[3]Drop time is the time (in seconds) after a broken connection has been detected until the connection is actually dropped. The drop time must be an even multiple of the probe time.

[4]Seconds.

[5]Trace is used to allow troubleshooting of the device communications. See Section 2.3.5 for details.

2.3.1.2 Editing BASEstar Open Protocol Profile Definitions

The BASEstar Open protocol profile is used to associate a device with the correct DAS. Example 2–2 shows the configuration file commands that create protocol profile definitions.

Example 2–2 BASEstar Open Protocol Profile Configuration

```
CREATE PROTOCOL_PROFILE mod_m84 \
 -APPLPROFID 101 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_184 \
 -APPLPROFID 102 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_384 \
 -APPLPROFID 103 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_484 \
 -APPLPROFID 104 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_584 \
 -APPLPROFID 105 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_884 \
 -APPLPROFID 106 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_984 \
 -APPLPROFID 107 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE modplus_984 \
 -APPLPROFID 108 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE modtcp_984 \
 -APPLPROFID 109 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
```

The DAS for Modicon programmable controllers uses a different Application Profile ID for each programmable controller family that it supports. When defining a VMD, be sure to specify the proper protocol profile for the family of programmable controller being defined.

The Modicon 984 family of devices uses different protocol profiles depending on the communication method used to communicate to the programmable controller.

Use the modplus_948 protocol profile for Modus Plus connections, mod_984 protocol profile for Modbus connections and modtcp_984 protocol profile for Modbus TCP/IP connections.

The Modicon 984 protocol profiles should be used when communicating with Quantum Automation Series programmable controllers.

The MAXPDUSIZE should be greater than or equal to the largest total packet sent by the device. If a packet is segmented into several smaller packets use the total packet size for this number.

2.3.1.3 Editing VMD Definitions

The VMD definition represents a Modicon device and references previously defined ODS and protocol profile definitions. Example 2–3 shows the configuration file commands that create VMD definitions.

Example 2–3 VMD Configuration

```
CREATE VMD mod_984 -MODEL "Modicon_984" \
                   -VENDOR "Modicon" \
                   -DVM_ACCESS_POINT "/PATH=MOD984_PATH":mod_984 \
                   -DESCRIPTION "Modicon 984 programmable controller" \
                   -LOG
```

(continued on next page)

Example 2–3 (Cont.) VMD Configuration

```
CREATE VMD modtcp_984 -MODEL "Modicon_984" \
                      -VENDOR "Modicon" \
                      -DVM_ACCESS_POINT "/PATH=MODTCP984_PATH":modtcp_984 \
                      -DESCRIPTION "Quantum Automation Series" \
                      -LOG
CREATE VMD modplus_984 -MODEL "Modicon_984" \
                      -VENDOR "Modicon" \
                      -DVM_ACCESS_POINT "/PATH=MODPLUS984_SLAVE_PATH":modplus_984 \
                      -DESCRIPTION "Modicon 984 programmable controller" \
                      -LOG
CREATE VMD mod_884 -MODEL "Modicon_884" \
                      -VENDOR "Modicon" \
                      -DVM_ACCESS_POINT "/PATH=MOD884_PATH":mod_884 \
                      -DESCRIPTION "Modicon 884 programmable controller" \
                      -LOG
CREATE VMD mod_584 -MODEL "Modicon_584" \
                      -VENDOR "Modicon" \
                      -DVM_ACCESS_POINT "/PATH=MOD584_PATH":mod_584 \
                      -DESCRIPTION "Modicon 584 programmable controller" \
                      -LOG
CREATE VMD mod_484 -MODEL "Modicon_484" \
                      -VENDOR "Modicon" \
                      -DVM_ACCESS_POINT "/PATH=MOD484_PATH":mod_484 \
                      -DESCRIPTION "Modicon 484 programmable controller" \
                      -LOG
```

An access point is created using the following syntax: `APPLICATION_SIMPLE_NAME:PROTOCOL_PROFILE_NAME`. The `APPLICATION_SIMPLE_NAME` is the definition registered in ODS. The `PROTOCOL_PROFILE_NAME` is defined using the BASEstar Open CLI as described in Section 2.3.1.2.

An access point tells the BASEstar Open software how to access the VMD. For more information on specifying client and DVM access points, refer to *BASEstar Open for OpenVMS Command Language Interface*.

Modify the command for your site configuration. Table 2–6 lists the modifiable device attributes.

Table 2–6 VMD CREATE

Parameter	Value(s)
MODEL[1]	Programmable controller model
VENDOR[1]	Modicon
REVISION[1]	
CLIENT_ACCESS_POINT[2]	
DVM_ACCESS_POINT[2]	
DESCRIPTION[1]	Descriptive text

[1]The value of this field is informational only and is not checked by the DAS.

[2]See the *BASEstar Open for OpenVMS Command Language Interface* for information on specifying client and DVM access points.

2.3.1.4 DAS Configuration Files

Example 2–4 and Example 2–5 are samples of the ODS and BSTR configuration files for the DAS for Modicon programmable controllers.

Example 2–4 ODS Configuration File

```
$ !
$ !*********************************************************************!
$ !                                                                     !
$ !    This is a template for the Modicon-specific                      !
$ !    ODS configuration command file.  Rename and edit this command    !
$ !    procedure template to reflect your site-specific                 !
$ !    configuration.  You must manually execute this command           !
$ !    procedure to configure ODS to provide line configuration         !
$ !    information for your devices.                                    !
$ !                                                                     !
$ !    Add comments to the site-specific changes here.                  !
$ !                                                                     !
$ !*********************************************************************!
$ !
$ODSCL := $ODS:[EXE]ODSCL.EXE
$ !
$ !  Register the ODS definitions.
$ !
$ODSCL
REGISTER DIRECTORY NAME "/LINE=MODBUS_LINE" ATTRIBUTES "OC=MODBUS_LINE
/PORT=LTA207:/RETRIES=3/SPEED=9600/PARITY=none/MODEM=0/TIMEOUT=3/TRACE=0"
REGISTER DIRECTORY NAME "/LINE=MODPLUS_LINE" ATTRIBUTES "OC=MODPLUS_LINE
/PORT=MPA1:/RETRIES=2/TIMEOUT=2/TRACE=0"

REGISTER DIRECTORY NAME "/PATH=MOD984_PATH" ATTRIBUTES "OC=MODBUS_PATH
/LINE_NAME=(/LINE=MODBUS)/SLAVE_ADDRESS=20"
REGISTER DIRECTORY NAME "/PATH=MODPLUS984_SLAVE_PATH" ATTRIBUTES
"OC=MODPLUS_SLAVE_PATH/LINE_NAME=(/LINE=MODPLUS_LINE)
/SLAVE_ROUTING=03.00.00.00.00/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MODPLUS984_NODE_PATH" ATTRIBUTES
"OC=MODPLUS_NODE_PATH/LINE_NAME=(/LINE=MODPLUS_LINE)/NODE_ADDRESS=4/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MOD884_PATH" ATTRIBUTES "OC=MODBUS_PATH
/LINE_NAME=(/LINE=MODBUS)/SLAVE_ADDRESS=21/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MOD584_PATH" ATTRIBUTES "OC=MODBUS_PATH
/LINE_NAME=(/LINE=MODBUS)/SLAVE_ADDRESS=22/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MOD484_PATH" ATTRIBUTES "OC=MODBUS_PATH
/LINE_NAME=(/LINE=MODBUS)/SLAVE_ADDRESS=23/TRACE=0"
REGISTER DIRECTORY NAME "/PATH=MODTCP984_PATH" ATTRIBUTES "OC=MODTCP_PATH
/IP_ADDRESS=19.20.20.5"
EXIT
```

To execute the ODS command file, enter the following command at the user prompt:

```
@ODS:[SCRIPTS]ODS_MOD_CONFIG.TEMPLATE
```

Example 2–5 BSTR Configuration File

```
!
!  Create the protocol profile definitions for the Modicon Micro84,
!  Modicon 184, Modicon 384,  Modicon 484, Modicon 584, Modicon 884,
!  and Modicon 984 family of programmable controllers.
!  Protocol profile IDs uniquely identify protocol support to OMNI.
!
CREATE PROTOCOL_PROFILE mod_m84 \
 -APPLPROFID 101 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_184 \
 -APPLPROFID 102 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_384 \
 -APPLPROFID 103 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_484 \
 -APPLPROFID 104 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_584 \
 -APPLPROFID 105 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_884 \
 -APPLPROFID 106 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE mod_984 \
 -APPLPROFID 107 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE modplus_984 \
 -APPLPROFID 108 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
CREATE PROTOCOL_PROFILE modtcp_984 \
 -APPLPROFID 109 -MAXPDUSIZE 1024 -SMA_MODE ENABLED -LOG
!
!  Create VMD definitions.
!  Use the ODS entries and protocol profile names created earlier.
!
CREATE VMD mod_984 -MODEL "Modicon_984" \
                   -VENDOR "Modicon" \
                   -DVM_ACCESS_POINT "/PATH=MOD984_PATH":mod_984 \
                   -DESCRIPTION "Modicon 984 programmable controller" \
                   -LOG
CREATE VMD modplus_984 -MODEL "Modicon_984" \
                   -VENDOR "Modicon" \
                   -DVM_ACCESS_POINT "/PATH=MODPLUS984_SLAVE_PATH":modplus_984 \
                   -DESCRIPTION "Modicon 984 programmable controller" \
                   -LOG
CREATE VMD modtcp_984 -MODEL "Modicon_984" \
                   -VENDOR "Modicon" \
                   -DVM_ACCESS_POINT "/PATH=MODTCP984_PATH":modtcp_984 \
                   -DESCRIPTION "Quantum Automation Series" \
                   -LOG
CREATE VMD mod_884 -MODEL "Modicon_884" \
                   -VENDOR "Modicon" \
                   -DVM_ACCESS_POINT "/PATH=MOD884_PATH":mod_884 \
                   -DESCRIPTION "Modicon 884 programmable controller" \
                   -LOG
CREATE VMD mod_584 -MODEL "Modicon_584" \
                   -VENDOR "Modicon" \
                   -DVM_ACCESS_POINT "/PATH=MOD584_PATH":mod_584 \
                   -DESCRIPTION "Modicon 584 programmable controller" \
                   -LOG
CREATE VMD mod_484 -MODEL "Modicon_484" \
                   -VENDOR "Modicon" \
                   -DVM_ACCESS_POINT "/PATH=MOD484_PATH":mod_484 \
                   -DESCRIPTION "Modicon 484 programmable controller" \
                   -LOG
```

(continued on next page)

Example 2–5 (Cont.) BSTR Configuration File

```
!
!  Create actors/activities for the VMDs
!
CREATE NODE mod_node -PHYSICAL "your_node" -LOG
CREATE DOMAIN /mod_984
CREATE DOMAIN /modplus_984
CREATE DOMAIN /modtcp_984
CREATE DOMAIN /mod_884
CREATE DOMAIN /mod_584
CREATE DOMAIN /mod_484
CREATE ACTOR /mod -LOG
SET CONTEXT /mod
CREATE PROGRAM mod_server -PROGRAM_KIND DATADEV -LOG
CREATE PROGRAM modplus_server -PROGRAM_KIND DATADEV -LOG
CREATE PROGRAM modtcp_server -PROGRAM_KIND DATADEV -LOG
CREATE ACTIVITY modbus_984 -PROGRAM mod_server \
                    -NODES (mod_node) \
                    -DOMAINS (/mod_984) \
                    -VMDS (mod_984) \
                    -LOG
CREATE ACTIVITY modplus_984 -PROGRAM modplus_server \
                    -NODES (mod_node) \
                    -DOMAINS (/modplus_984) \
                    -VMDS (modplus_984) \
                    -LOG
CREATE ACTIVITY modtcp_984 -PROGRAM modtcp_server \
                    -NODES (mod_node) \
      -DOMAINS (/modtcp_984) \
                    -VMDS (modtcp_984) \
                    -LOG
CREATE ACTIVITY modbus_884 -PROGRAM mod_server \
                    -NODES (mod_node) \
                    -DOMAINS (/mod_884) \
                    -VMDS (mod_884) \
                    -LOG
CREATE ACTIVITY modbus_584 -PROGRAM mod_server \
                    -NODES (mod_node) \
                    -DOMAINS (/mod_584) \
                    -VMDS (mod_584) \
                    -LOG
CREATE ACTIVITY modbus_484 -PROGRAM mod_server \
                    -NODES (mod_node) \
                    -DOMAINS (/mod_484) \
                    -VMDS (mod_484) \
                    -LOG
```

To execute the BASEstar Open command file, enter the following command at the user prompt:

```
$CLI_BSTR -F BSTR_MOD_CONFIG.TEMPLATE
```

For more information about maintaining protocol profile and device definitions, refer to the *BASEstar Open for OpenVMS Command Language Interface*. For more information about maintaining ODS definitions, refer to the *Compaq OMNI for OpenVMS Guide to Using Omni Directory Services*.

2.3.2 Configuring Ports

You can physically connect a device to a Local Area Transport (LAT) port. To use a LAT port with the DAS, you must define the LAT port on the OpenVMS system using the LAT control program.

For example, to define **port_2** on the LAT node **srvr4** as **LTA992**, enter the following:

```
$ MCR LATCP
LCP> CREATE PORT LTA992
LCP> SET PORT LTA992 /NODE=srvr4/PORT=port_2/NOQUEUE
```

Use the LAT port name defined above when defining the port to ODS.

You must also set the following SYSGEN parameters to access Modicon programmable controllers to reduce the potential for data overruns. Use the SYSGEN utility to display the current values and AUTOGEN to reset the values accordingly. For information on using SYSGEN and AUTOGEN, refer to the *OpenVMS System Management Utilities Reference Manual* for details.

Parameter	Value
TTY_ALTALARM	0
TTY_ALTYPAHD	512

To enable use of the alternate typeahead buffer, type the following command from DCL:

```
$ SET TERM LTA992:/ALTYPEAHD/PERMANENT
```

For the DAS for Modicon programmable controllers to send and receive data correctly, you must set and define LAT port characteristics to match the line parameters defined in ODS for the corresponding port. The following example shows the characteristics set at the LAT prompt:

```
Local> DEF PORT n SPEED 9600
Local> DEF PORT n AUTOBAUD DISABLED
Local> DEF PORT n AUTOCONNECT DISABLED
Local> DEF PORT n FLOW DISABLED
Local> DEF PORT n PREFERRED NONE
Local> DEF PORT n ACCESS REMOTE
Local> DEF PORT n CHAR 8
Local> DEF PORT n PARITY EVEN
```

2.3.3 Configuring TELNET Access

You can physically connect a programmable controller to a terminal server using the TELNET protocol. Using the TELNET protocol allows the use of non-proprietary protocols to access terminal servers to perform device communications.

The terminal server being used must support the BINARY transfer option (RFC 856). The DAS negotiates binary transfer as part of establishing a connection with the port. If the terminal server will not allow binary transfer, the negotiation fails and a connection is not established.

NOTE: Using TELNET communications is slower than either LAT or a directly connected device. Ensure that your throughput requirements will be satisfied before deciding to implement TELNET communications.

For the DAS for Modicon programmable controllers to send and receive data correctly using TELNET, you must set up a port on the terminal server to receive a connect from the DAS. The following example shows setting up a port using a DECserver 700.

```
Local> DEF TELNET LISTENER y PORT n
Local> DEF TELNET LISTENER y CONNECTIONS ENABLED
```

The following example shows setting up the server for TELNET access. A gateway needs to be configured only if the terminal server is on a different subnet than the host. In general, the terminal server should be on the same subnet as the host in order to reduce communications delays.

```
Local> DEF INTERNET ADDRESS xx.xx.xx.xx
Local> DEF INTERNET MASK xx.xx.xx.xx
Local> DEF INTERNET GATEWAY xx.xx.xx.xx
```

2.3.4 Setting Up Plant-Floor Equipment

To set up your plant-floor equipment, refer to the Modicon documentation for your specific device.

2.3.5 Tracing Device Communications

The DAS for Modicon programmable controllers has built into it the capablity to trace device communications and to send this output to either a file or to a user terminal. The purpose of this tracing is to allow the troubleshooting of device communications.

Tracing can be enabled for a device and/or a line. If tracing is enabled for a device, then only communications from/to that device are traced. If tracing is enabled for a line, then all communications from/to that line are traced. If both device and line tracing is enabled, then the trace will contain the output for both the device and the line, so the trace output will be duplicated.

To enable tracing do the following:

- Set the TRACE attribute in the ODS definition for the path or line being traced.
- Set the DAS_TRACE_OUTPUT logical name to the name of the file that you want to contain the trace output. The logical name must be defined in a scope that will be seen by the server the DAS is a part of.

  ```
  DEFINE DAS_TRACE_OUTPUT TRACE_OUTPUT.LOG
  ```

- Reinitiate communications to the device/line. The ODS definition is read when a connection to the device is initiated (device tracing) or when a connection to the line is initiated (line tracing), so communications must be reinitiated in order for tracing to occur.

The trace output is flushed to the file after every 10 lines that are traced, so trace output may lag actual device I/O.

All trace output is in hexadecimal and shows both input to and output from the device.

To turn tracing off, reverse the steps used to turn tracing on.

2.4 Failures During Product Use

If an error occurs while this product is in use and you believe the error is caused by a problem with the product, take one of the following actions:

- If you have a Software Product Services Support Agreement, contact your Customer Support Center (CSC) by telephone or by using the electronic means provided with your support agreement (such as DSNlink). The CSC provides telephone support for high-level advisory and remedial assistance. When you initially contact the CSC, indicate the following:
  - The name and version number of the operating system you are using
  - The version number of the product you are using
  - The version number of BASEstar Open you are using
  - The hardware system you are using (such as a model number)
  - The Modicon programmable controllers you are communicating with
  - A brief description of the problem (one sentence if possible)
  - How critical the problem is
- If you have a Self-Maintenance Software Agreement, you can submit a Software Performance Report (SPR).
- If you do not have any type of software services support agreement and you purchased this product within the past year, you can submit an SPR if you think the problem is caused by a software error.

When you submit an SPR, take the following steps:

1. Describe as accurately as possible the circumstances and state of the system when the problem occurred. Include the description and version number of the product being used. Demonstrate the problem with specific examples.
2. Reduce the problem to as small a size as possible.
3. Remember to include listings of any command files, INCLUDE files, or relevant data files, and so forth.
4. Report only one problem per SPR. This will facilitate a faster response.
5. Mail the SPR package to Compaq.

3
Using the DAS

This chapter provides information about the supported functions for Modicon devices, and how to access these functions.

3.1 Accessing Modicon Functions

DAS for Modicon programmable controllers functions are accessed through the following interfaces:

- BASEstar Open Command Language Interface (CLI)
- BASEstar Open Application Programming Interface (API)
- OMNI Application Programming Interface (API)

Command Language Interface (CLI) To access the BASEstar Open CLI, enter the following command at the DCL prompt ($):

```
$ CLI_BSTR
```

For information about BASEstar Open commands, refer to the *BASEstar Open for OpenVMS Command Language Interface*.

Application Programming Interface (API) For information on accessing OMNI Callable Services, refer to the *Compaq OMNI for OpenVMS Application Programmer's Interface*.

3.2 Supported Functions

A variety of functions are supported by the DAS for Modicon programmable controllers. These functions may be supported as either client or server or both. If a function is supported as a server, then the application will receive an *indication* when the VMD is requesting the service.

The DAS for Modicon programmable controllers supports the following client functions for all supported devices:

- Connect
- Read Variable
- Write Variable
- Abort
- Conclude

The DAS for Modicon programmable controllers supports the following server functions (indications) for all supported devices:

- Write indication
- Abort

3.2.1 Connect, Conclude and Abort Functions

Table 3–1 lists the connect, conclude and abort functions available through the OMNI (API) interface.

Table 3–1 Connect, Conclude and Abort Interface Access

Interface	Function Name
API	omni_connect
	omni_connect_a
	omni_conclude
	omni_conclude_a
	omni_abort
	omni_abort_a

3.2.2 Read and Write Variable Functions

Use the read variable and write variable functions to read data from and write data to a specific address in device memory.

Table 3–2 lists the variable types supported by this DAS.

Table 3–2 Read and Write Variable Types

Variable Type	Address Type	Supported
UNNAMED_VARIABLE	NUMERIC	Yes
	SYMBOLIC	No
	UNCONSTRAINED	Yes[1]
NAMED_VARIABLE		Yes[2]

[1]Unconstrained addresses must be used for Modicon 584 and 984 extended addresses. Unconstrained addresses must also be used for Modicon 984 coil addresses of 010000 or higher to distinguish them from discrete input addresses.

[2]Named variables are not supported by the DAS, but they are used by BASEstar Open when accessing variables using the Device Manager Server. The BASEstar Open Device Manager Server converts named variables with an address type of UNCONSTRAINED to unnamed variables.

An UNNAMED_VARIABLE type describes a specific physical address on a VMD. This address has a predefined type based on the address. Variables of this type are an inherent part of the VMD architecture and are uniquely identified by their physical address.

An unnamed variable can be one of the following types:

- NUMERIC

  A numeric address must be an MMS unsigned integer.

- SYMBOLIC

  A symbolic address must be an MMS visible string.

- UNCONSTRAINED

  An unconstrained address can be any MMS octet (byte) string.

A NAMED_VARIABLE type describes a variable on a device. Variables of this type are uniquely identified by their name.

Table 3–3 lists the read and write functions available through the BASEstar Open (CLI) and OMNI (API) interfaces.

Table 3–3 Read and Write Variable Interface Access

Interface	Function Name
CLI	GET VALUE UNNAMED_VARIABLE
	GET VALUE NAMED_VARIABLE
	PUT VALUE UNNAMED_VARIABLE
	PUT VALUE NAMED_VARIABLE
API	omni_get_value
	omni_get_value_a
	omni_put_value
	omni_put_value_a

Example 3–1 shows examples of creating unnamed variables using the CREATE UNNAMED_VARIABLE command.

Example 3–1 UNNAMED_VARIABLE CREATE Command

```
BSTR> CREATE UNNAMED_VARIABLE mod_184.p1 \
      -DATATYPE INTEGER_16 \
      -ADDRESS "4001" \
      -ADDRTYPE NUMERIC
BSTR> CREATE UNNAMED_VARIABLE mod_384.p1 \
      -DATATYPE INTEGER_16 \
      -ADDRESS "4001" \
      -ADDRTYPE NUMERIC
BSTR> CREATE UNNAMED_VARIABLE mod_584.p1 \
      -DATATYPE BIT \
      -ADDRESS "1" \
      -ADDRTYPE NUMERIC
BSTR> CREATE ARRAY_DATATYPE WORD_ARRAY_10 \
      -DATATYPE INTEGER_16 \
      -SIZE 10
```

(continued on next page)

Example 3–1 (Cont.) UNNAMED_VARIABLE CREATE Command

```
BSTR> CREATE UNNAMED_VARIABLE mod_584.p2 \
      -DATATYPE WORD_ARRAY_10 \
      -ADDRESS "40001" \
      -ADDRTYPE NUMERIC
BSTR> CREATE UNNAMED_VARIABLE mod_884.p1 \
      -DATATYPE BIT \
      -ADDRESS "10001" \
      -ADDRTYPE NUMERIC
BSTR> CREATE UNNAMED_VARIABLE mod_984.p1 \
      -DATATYPE INTEGER_16 \
      -ADDRESS "60001:1" \
      -ADDRTYPE UNCONSTRAINED
BSTR> CREATE UNNAMED_VARIABLE mod_984.p2 \
      -DATATYPE INTEGER_16 \
      -ADDRESS "40001" \
      -ADDRTYPE NUMERIC
```

Example 3–2 shows the information displayed when you enter the GET VALUE UNNAMED_VARIABLE command.

Example 3–2 UNNAMED_VARIABLE GET VALUE Output

```
BSTR> GET VALUE UNNAMED_VARIABLE mod_184.p1

20
BSTR> GET VALUE UNNAMED_VARIABLE mod_384.p1

10
BSTR> GET VALUE UNNAMED_VARIABLE mod_584.p1

0
BSTR> GET VALUE UNNAMED_VARIABLE mod_584.p2

array[10] of INTEGER_16
  [ 0]: 0
  [ 1]: 0
  [ 2]: 0
  [ 3]: 0
  [ 4]: 0
  [ 5]: 0
  [ 6]: 0
  [ 7]: 0
  [ 8]: 0
  [ 9]: 0
BSTR> GET VALUE UNNAMED_VARIABLE mod_884.p1

1
BSTR> GET VALUE UNNAMED_VARIABLE mod_984.p1

30
BSTR> GET VALUE UNNAMED_VARIABLE mod_984.p2

40
```

Example 3–3 shows the information displayed when you enter the PUT VALUE UNNAMED_VARIABLE command.

Example 3–3 UNNAMED_VARIABLE PUT VALUE Output

```
BSTR> PUT VALUE UNNAMED_VARIABLE mod_584.p1

(Bit) : 1
BSTR> PUT VALUE UNNAMED_VARIABLE mod_584.p2

array[10] of INTEGER_16
  [ 0]: (Integer 16) : 1
  [ 1]: (Integer 16) : 2
  [ 2]: (Integer 16) : 3
  [ 3]: (Integer 16) : 4
  [ 4]: (Integer 16) : 5
  [ 5]: (Integer 16) : 6
  [ 6]: (Integer 16) : 7
  [ 7]: (Integer 16) : 8
  [ 8]: (Integer 16) : 9
  [ 9]: (Integer 16) : 10
```

3.2.3 Read and Write Data Point Functions

Use the read data point and write data point functions to read data from and write data to a device in a device independent way.

Table 3–4 lists the read and write data point functions available through the BASEstar Open (CLI) interface and BASEstar Open (API) interface.

Table 3–4 Read and Write Data Point Interface Access

Interface	Function Name
CLI	GET VALUE DATA_POINT
	PUT VALUE DATA_POINT
API	bstr_data_point_get_value
	bstr_data_point_put_value

Example 3–4 shows examples of creating a device data point using the CREATE DEVICE_DATA_POINT command.

Example 3–4 DEVICE_DATA_POINT CREATE Command

```
BSTR> CREATE DEVICE_DATA_POINT /mod_584/p1 \
                   -DATATYPE BIT \
                   -UPDATE_POLICY SOLICITED \
                   -ACCESS_POLICY ALWAYS \
                   -DEVICE_ACCESS READ \
                   -VMD_NAME mod_584 \
                   -VAR_NAME p1 \
                   -VAR_CLASS UNNAMED
```

(continued on next page)

Example 3–4 (Cont.) DEVICE_DATA_POINT CREATE Command

```
BSTR> CREATE DEVICE_DATA_POINT /mod_584/p2 \
                    -DATATYPE WORD_ARRAY_10 \
                    -UPDATE_POLICY SOLICITED \
                    -ACCESS_POLICY ALWAYS \
                    -DEVICE_ACCESS RDWR \
                    -VMD_NAME mod_584 \
                    -VAR_NAME p2 \
                    -VAR_CLASS UNNAMED
BSTR> CREATE DEVICE_DATA_POINT /mod_984/p1 \
                    -DATATYPE INTEGER_16 \
                    -UPDATE_POLICY SOLICITED \
                    -ACCESS_POLICY ALWAYS \
                    -DEVICE_ACCESS RDWR \
                    -VMD_NAME mod_984 \
                    -VAR_NAME p1 \
                    -VAR_CLASS UNNAMED
BSTR> CREATE DEVICE_DATA_POINT /mod_984/p2 \
                    -DATATYPE INTEGER_16 \
                    -UPDATE_POLICY SOLICITED \
                    -ACCESS_POLICY ALWAYS \
                    -DEVICE_ACCESS RDWR \
                    -VMD_NAME mod_984 \
                    -VAR_NAME p2 \
                    -VAR_CLASS UNNAMED
```

Example 3–5 shows the information displayed when you enter the GET VALUE DATA_POINT command.

Example 3–5 DATA_POINT GET VALUE Output

```
BSTR> GET VALUE DATA_POINT /mod_584/p1

1
BSTR> GET VALUE DATA_POINT /mod_584/p2

array[10] of INTEGER_16
  [ 0]: 0
  [ 1]: 0
  [ 2]: 0
  [ 3]: 0
  [ 4]: 0
  [ 5]: 0
  [ 6]: 0
  [ 7]: 0
  [ 8]: 0
  [ 9]: 0
```

Example 3–6 shows the information displayed when you enter the PUT VALUE DATA_POINT command.

Example 3–6 DATA_POINT PUT VALUE Output

```
BSTR> PUT VALUE DATA_POINT /mod_584/p1

(Bit) : 1
BSTR> PUT VALUE DATA_POINT /mod_584/p2

array[10] of INTEGER_16
  [ 0]: (Integer 16) : 1
  [ 1]: (Integer 16) : 2
  [ 2]: (Integer 16) : 3
  [ 3]: (Integer 16) : 4
  [ 4]: (Integer 16) : 5
  [ 5]: (Integer 16) : 6
  [ 6]: (Integer 16) : 7
  [ 7]: (Integer 16) : 8
  [ 8]: (Integer 16) : 9
  [ 9]: (Integer 16) : 10
```

3.2.4 Receiving Indications

An indication is data that is sent to the DAS from the programmable controller without the DAS specifically requesting the data. The DAS supports write and abort indications.

Table 3–5 lists the indication functions available through the OMNI (API) interface.

Table 3–5 Indication Interface Access

Interface	Function Name
API	omni_get_indications
	omni_get_indications_a

3.2.4.1 Write Indications

The write indication is used when the device is sending data to the host. Write indications are only supported for Modbus Plus and Modbus TCP/IP connections which support peer-to-peer communications. With Modbus Plus and Modbus TCP/IP, the host can receive slave commands as well as issue master commands.

In order to use the write indication function, a variable must be created on the host that matches the address and size of the data being sent by the programmable controller.

If a programmable controller has been programmed to send data to the host and a variable is found that matches the address of the data being sent, a write indication is generated specifying the variable that was found. If more than one variable is configured that matches the address of the data being sent, then an indication is generated only for the first variable found.

When programming the programmable controller to send data to the host, use the special master function (MSTR) block with the write data function code (1).

When using Modbus Plus any slave path that has been configured on the MODPLUS_LINE can be used when sending data to the host. See Table 2–4 for information on how to configure the number of slave paths available.

The DAS supports writing from the programmable controller to the host only. Reads from the programmable controller to the host are not currently supported.

Example 3–7 shows an example of how to configure VMDs to receive write indications from a Modicon 984 with a Modbus Plus connection. The Modicon 984 has been configured to send messages to the host. A Modicon 984 is configured in BASEstar Open to receive the write indications.

Note

The path specified for the local VMD uses a different ODS class than the path for specifying the remote VMD. This difference is only true when using programmable controllers with a Modbus Plus connection. See Table 2–4 for details on how to configure this ODS Modbus Plus class. The path on the local 984 is used to obtain the node number of the SQ /SA-85 card which is then used as one of the parameters used to read data from the slave path.

Example 3–7 Modbus Plus Write Indication Configuration

```
CREATE PROTOCOL_PROFILE modplus_984 \
                    -APPLPROFID 108 \
                    -MAXPDUSIZE 1024 \
                    -SMA_MODE ENABLED \
                    -LOG
CREATE VMD mod_984_remote \
                    -MODEL "Modicon 984" \
                    -VENDOR "Modicon" \
                    -DVM_ACCESS_POINT "/PATH=MODPLUS984_SLAVE_PATH":modplus_984 \
                    -DESCRIPTION "Modicon 984 programmable controller" \
                    -LOG
CREATE VMD mod_984_local \
                    -MODEL "Modicon 984" \
                    -VENDOR "Modicon" \
                    -DVM_ACCESS_POINT "/PATH=MODPLUS984_NODE_PATH":modplus_984 \
                    -DESCRIPTION "Modicon SQ/SA-85 card" \
                    -LOG
```

Example 3–8 shows an example of how to configure VMDs to receive write indications from a Modicon 984 with a Modbus TCP/IP connection. The Modicon 984 has been configured to send messages to the host. A Modicon 984 is configured in BASEstar Open to receive the write indications.

Note

The path on the local VMD is not used since the VMD is only used to receive the write indications and is not used in establishing communications to the PLC.

Example 3–8 Modbus TCP/IP Write Indication Configuration

```
CREATE PROTOCOL_PROFILE modtcp_984 \
                    -APPLPROFID 109 \
                    -MAXPDUSIZE 1024 \
                    -SMA_MODE ENABLED \
                    -LOG
CREATE VMD mod_984_remote \
                    -MODEL "Modicon 984" \
                    -VENDOR "Modicon" \
                    -DVM_ACCESS_POINT "/PATH=MODTCP984_PATH":modtcp_984 \
                    -DESCRIPTION "Modicon 984 programmable controller" \
                    -LOG
CREATE VMD mod_984_local \
                    -MODEL "Modicon 984" \
                    -VENDOR "Modicon" \
                    -DVM_ACCESS_POINT "/PATH=MODTCP984_PATH":modtcp_984 \
                    -DESCRIPTION "Modicon 984 programmable controller" \
                    -LOG
```

Example 3–9 shows an example of how to configure actors and activities to receive write indications from a Modicon 984 that has been configured above.

Example 3–9 Modbus Write Indication Configuration

```
!
!  Create actors/activities for the VMDs
!
CREATE NODE mod_node -PHYSICAL "your_node" -LOG
CREATE DOMAIN /mod_data
CREATE ACTOR /MOD_IND -LOG
SET CONTEXT /MOD_IND
CREATE PROGRAM mod_data_srv -PROGRAM_KIND DATADEV \
                    -PARAM MOD_PARAM:MOD.PAR \
                    -LOG
CREATE_PROGRAM mod_event_srv -PROGRAM_KIND EVENT \
                    -LOG
CREATE ACTIVITY mod_data_act -PROGRAM mod_data_srv \
                    -NODES (mod_node) \
                    -DOMAINS (/mod_data) \
                    -VMDS (mod_984_remote) \
                    -LOG
CREATE ACTIVITY mod_event_act -PROGRAM mod_event_srv \
                    -NODES (mod_node) \
                    -DOMAINS (/mod_data) \
                    -LOG
```

The CREATE PROGRAM command shown in Example 3–9 has an additional attribute, the -PARAM attribute, that specifies a parameter file to be read by the mod_data_srv program. The parameter file contains the name of the local VMD to receive the write indications as shown in Example 3–10. See *BASEstar Open for OpenVMS Management Guide* for more information on possible contents of the parameter file.

Example 3–10 MOD.PAR Contents

```
/MOD_IND/mod_data_srv.env: BSTR_LOCAL_VMD_NAME=mod_984_local
```

Only one parameter file can be specified per program so BASEstar Open limits the number of local VMDs that can be specified to one per program. This limitation means that **all** write indications for a particular program must be directed to one local VMD, so all programmable controllers connected to that line must use unique addresses to ensure that the data from the programmable controller is uniquely identified.

Example 3–11 shows an example that creates variables, device data points, events, triggers and enboxes to receive write indications from the Modicon 984 configured in Example 3–9.

Example 3–11 Write Indication Example

```
BSTR> EXECUTE ACTOR /MOD_IND -WAIT
BSTR> CREATE UNNAMED_VARIABLE mod_984_local.p1 \
      -DATATYPE UNSIGNED_16 \
      -ADDRESS "40001" \
      -ADDRTYPE UNCONSTRAINED
BSTR> CREATE DEVICE_DATA_POINT /mod_data/p1 \
                    -DATATYPE UNSIGNED_16 \
                    -UPDATE_POLICY UNSOLICITED \
                    -ACCESS_POLICY ON_REQUEST \
                    -DEVICE_ACCESS READ \
                    -VMD_NAME mod_984_local \
                    -VAR_NAME p1 \
                    -VAR_CLASS UNNAMED
BSTR> CREATE ENBOX /mod_data/enbox1 \
                    -QUEUE_KIND SEQUENTIAL \
                    -ENBOX_KIND SYSTEM_BUFFERING \
                    -MAX_NOTIFICATIONS 10
BSTR> CREATE EVENT /mod_data/e1 \
                    -PARAMETER_TYPES (INTEGER_16)
BSTR> CREATE TRIGGER /mod_data/t1 \
                    -STATE ENABLED \
                    -TRIGGER_EVALUATION ON_WRITE \
                    -DATA_POINT /mod_data/p1 \
                    -EVENT /ab_data/e1 \
                    -EVE_PARAMETERS (/mod_data/p1:VALUE)
BSTR> ADD SUBSCRIPTION EVENT /mod_data/e1 /mod_data/enbox1 \
                    -NOTIFICATION PARAMETERS_LIST \
                    -STATE ENABLED
BSTR> CONNECT ENBOX /mod_data/enbox1
BSTR> RECEIVE NOTIFICATION ENBOX /mod_data/enbox1
Event              : /mod_data/e1
Origin             : /mod_data/t1
Priority           : 0
Confirmation       : NOT_CONFIRMED
Notification_id    : 1
User_parameter     : 0
Notification_format: PARAMETERS_LIST
Time               : Thu Mar 23 08:26:50 2000

Parameter 1 value:
1
BSTR> DISCONNECT ENBOX /mod_data/enbox1
BSTR> REMOVE SUBSCRIPTION EVENT /mod_data/e1 /mod_data/enbox1
```

The use of the value INIT_UNSOLICITED for the -UPDATE_POLICY qualifier is not supported for write indications since the address of the variable in the remote VMD is not available. If an application requires the value of an unsolicited variable when starting, before the value has been updated by the device, create a

device data point and unnamed variable on the remote VMD and use that to read the initial value.

If a programmable controller has been programmed to send data to the host and a variable is found that matches the address of the data being sent, a write indication is generated specifying the variable that was found. If more than one variable is configured that matches the address of the data being sent, then an indication is generated for only the first variable found.

3.2.4.2 Abort Indications

An abort indication is generated by the DAS when it loses a connection to the device. The DAS detects that a connection is broken in different ways depending on the type of connection to the device.

For an RS-232 connection, the DAS aborts the connection when an error other than a time timeout is returned from a QIO call or if retries are exhausted. In these cases all connections using this port are aborted. The DAS aborts the connection to an individual device if there is a timeout.

For a Modbus Plus connection, the DAS aborts the connection when any QIO error occurs when reading from the device. In this case all connections using this port are aborted.

For a Modbus TCP/IP connection, the DAS aborts the connection when any I/O error occurs when reading from the device.

3.3 Addressing Syntax and Supported Data Types

Data types are defined at the device and application levels. The device data type is referred to as the *MMS data type*. The MMS data type is mapped to an application data type.

Data types can be either *simple* or *complex*. Simple data types are native to the OMNI environment. Complex data types are divided into either arrays or structures. Complex data types must be created by a user before they can be used in defining data points or variables. Structures are not supported for the DAS for Modicon programmable controllers.

The following sections describe the supported data types and addressing syntax for each of the supported device families.

3.3.1 Modicon Micro 84 Addressing Syntax and Supported Data Types

Table 3–6 shows the structure of a legal address for the Micro 84 family.

Table 3–6 Modicon Micro 84 Address Format

Address Format	MMS Type[1]	Description
0001 - 0064	BIT	Coils
1001 - 1064[2]	BIT	Discrete inputs
3001 - 3004[2]	UNSIGNED_16 INTEGER_16	Input registers
4001 - 4032	UNSIGNED_16 INTEGER_16	Holding registers

[1]Arrays are supported. Variable size arrays are **not** supported.

[2]Writes are **not** supported.

3.3.2 Modicon 184 and 384 Addressing Syntax and Supported Data Types

Table 3–7 shows the structure of a legal address for the Modicon 184 and 384 families.

Table 3–7 Modicon 184 and 384 Address Format

Address Format	MMS Type[1]	Description
0001 - 0992	BIT	Coils
1001 - 1992[2]	BIT	Discrete inputs
3001 - 3999[2]	UNSIGNED_16 INTEGER_16	Input registers
4001 - 4999	UNSIGNED_16 INTEGER_16	Holding registers

[1]Arrays are supported. Variable size arrays are **not** supported.

[2]Writes are **not** supported.

3.3.3 Modicon 484 Addressing Syntax and Supported Data Types

Table 3–8 shows the structure of a legal address for the Modicon 484 family.

Table 3–8 Modicon 484 Address Format

Address Format	MMS Type[1]	Description
0001 - 0512	BIT	Coils
1001 - 1256[2]	BIT	Discrete inputs
3001 - 3032[2]	UNSIGNED_16 INTEGER_16	Input registers
4001 - 4254	UNSIGNED_16 INTEGER_16	Holding registers

[1]Arrays are supported. Variable size arrays are **not** supported.

[2]Writes are **not** supported.

3.3.4 Modicon 884 Addressing Syntax and Supported Data Types

Table 3–9 shows the structure of a legal address for the Modicon 884 family.

Table 3–9 Modicon 884 Address Format

Address Format	MMS Type[1]	Description[3]
00001 - 00768	BIT	Coils
10001 - 10256[2]	BIT	Discrete inputs
30001 - 30256[2]	UNSIGNED_16 INTEGER_16	Input registers
40001 - 40864	UNSIGNED_16 INTEGER_16	Holding registers

[1]Arrays are supported. Variable size arrays are **not** supported.

[2]Writes are **not** supported.

[3]Sequencer registers, double precision registers and latches are not supported.

3.3.5 Modicon 584 and 984 Addressing Syntax and Supported Data Types

Table 3–10 shows the structure of a legal address for the Modicon 584 and 984 families.

Table 3–10 Modicon 584 and 984 Address Format

Address Format	MMS Type[1]	Description
00001 - 08192	BIT	Coils
10001 - 18192[2]	BIT	Discrete inputs
30001 - 39999[2]	UNSIGNED_16 INTEGER_16	Input registers
40001 - 49999	UNSIGNED_16 INTEGER_16	Holding registers
60000:1 - 69999:10[3]	UNSIGNED_16 INTEGER_16	Extended file
000001 - 065535[4]	BIT	Coils
100001 - 116384[2]	BIT	Discrete inputs
300001 - 365535[2]	UNSIGNED_16 INTEGER_16	Input registers
400001 - 465535	UNSIGNED_16 INTEGER_16 FLOAT	Holding registers
600000:1 - 665535:10[3]	UNSIGNED_16 INTEGER_16	Extended file

[1] Arrays are supported. Variable size arrays are **not** supported.

[2] Writes are **not** supported.

[3] If a file number is not specified, then the file number defaults to 1.

[4] Six digit addresses are supported for Modicon 984's only. To specify a coil 010000 or above, use an unconstrained address, not a numeric address, in order to distinguish the address from a discrete input.

A

I/O Error Messages

This appendix contains returned values and associated error messages for the low level I/O routines.

allocerror, An error occurred when allocating an OpenVMS port,

Explanation: Error. An error occurred when allocating an OpenVMS port. The allocation is done when a connection is initiated with the VMD.

User Action: An error message detailing the error should have been logged. Examine the error to determine the proper corrective action.

allocstatus, Status for port XX returned from SYS$ALLOC, XX,

Explanation: Error. An error occurred when allocating a port.

User Action: Examine the error message to determine the cause of the error and take appropriate action based on the error message.

assignerror, An error occurred when assigning a channel to an OpenVMS port,

Explanation: Error. An error occurred when assigning a channel to an OpenVMS port. The assignment is done when a connection is initiated with the VMD.

User Action: An error message detailing the error should have been logged. Examine the error to determine the proper corrective action.

assignstatus, Status for port XX returned from SYS$ASSIGN, XX,

Explanation: Error. An error occurred when assigning a channel.

User Action: Examine the error message to determine the cause of the error and take appropriate action based on the error message.

connecterror, Error connecting to Internet address,

Explanation: Error. An error occurred when attempting to connect to an Internet address.

User Action: Examine the log file (BASEstar Open) or stderr to determine the error that occurred. Then take corrective action based on the error that was found.

dataoverrun, Data overrun occurred on the line,

Explanation: Error. A data overrun occurred on the line.

User Action: Use a slower baud rate or switch to a port that has faster hardware.

dont, TELNET server refused to allow TELNET option,

Explanation: Error. The TELNET server responded with "DON'T" to a request, "WILL", for the requested TELNET option.

User Action: Enable the apropriate option on the TELNET server or substitute a server that supports requested option.

dontdobin, TELNET server refused to allow binary data reception,

Explanation: Error. The TELNET server responded with "DON'T" to a request, "WILL", for the host to send binary data (TELNET option RFC 856).

User Action: Enable binary mode transmission on the TELNET server or substitute a server that supports binary mode transmission/reception.

econnrefused, The attempt to connect was rejected for port XX,

Explanation: Error. An attempt to connect resulted was rejected by the remote server.

User Action: If using TELNET, check the terminal server to ensure that the listener port has been properly configured and enabled.

ehostunreach, No route to host is present for port XX,

Explanation: Error. An attempt to connect to a TELNET port resulted in an unreachable status being returned.

User Action: Make sure that the IP address and port number are properly defined and that a gateway is defined on the host.

enetunreach, No route to the network or host is present for port XX,

Explanation: Error. An attempt to connect to a TELNET port resulted in an unreachable status being returned.

User Action: Make sure that the IP address and port number are properly defined and that a gateway is defined on the host.

etimedout, Connection timed out for port XX,

Explanation: Error. An attempt to connect to a port or socket timed out.

User Action: If using LAT, make sure that the port is defined in the LAT Control Program and that noone else is using the port. Then attempt to reconnect to the port. If using TELNET, make sure that the IP address and port number are defined properly on the terminal server and that the listener port is enabled. Also ensure that a gateway is defined if the host and the terminal server are not on the same subnet.

getdvistatus, Status for port XX returned from SYS$GETDVI, XX,

Explanation: Error. An error occurred when getting device information.

User Action: Examine the error message and the port definition to determine the cause of the error and take appropriate action based on the error message.

inaddrerror, Error when converting an Internet address to binary form,

Explanation: Error. An error occurred when convertng an Internet address

User Action: Examine the Internet address to determine the error and then correct the Internet address.

invalidport, An invalid port definition was specified,

Explanation: Error. A port definition that was defined in ODS is not valid. Either the port is not a valid TT type device or device characteristics could not be obtained for the port.

User Action: Examine the log file (BASEstar) or stderr to determine the error status and the function which detected the error. Take action based on the error message.

invalidspeed, An invalid baud rate was specified,

Explanation: Error. A baud rate that was defined in ODS is not valid.

User Action: Check the documentation to determine valid baud rates for the hardware platform being used. Then modify the baud rate in ODS to a valid value.

latcode, Status for port XX returned from SYS$QIOW (IO$_TTY_PORT | IO$M_LT_CONNECT) = X,

Explanation: Error. An error occurred when connecting to a LAT port.

User Action: Examine the error code to determine the cause of the error and take appropriate action based on the error code.

laterror, An error occurred when connecting to a LAT port,

Explanation: Error. An error occurred when connecting to a LAT port. The connection is done when a connection is initiated with the VMD.

User Action: An error message detailing the error should have been logged. Examine the error to determine the proper corrective action.

latmodeminv, Invalid attempt to enable modem signals on a LAT port,

Explanation: Error. LAT ports do not support setting of the "modem" ODS parameter to "TRUE".

User Action: Change the port to a non-LAT port or change the "modem" parameter to "FALSE", "0" or "OFF".

latstatus, Status for port XX returned from SYS$QIOW (IO$_TTY_PORT | IO$M_LT_CONNECT), XX,

Explanation: Error. An error occurred when connecting to a LAT port.

User Action: Examine the error message to determine the cause of the error and take appropriate action based on the error message.

negfailed, TELNET option negotiation failed,

Explanation: Error. An I/O error occurred when attempting to negotiate TELNET options.

User Action: Examine the log file (BASEstar Open) or stderr to determine the cause of the error and then take corrective action based on the error found.

noalttypeahead, "/ALTYPEAHD" must be specified for the given port,

Explanation: Error. The DAS requires that /ALTYPEAHD be specified for the port that is being used to communicate to the device. ALTYPEAHD helps to ensure that characters transmitted by the device are not lost when the host is busy.

User Action: Use DCL to "SET TERM/PERM/ALTYPEAHD port_name" on the port being used to communicate to the device(s).

nolicense, No software license,

Explanation: Error. The DAS requires a software license.

User Action: Install the license PAK for the DAS.

parity, Parity error,

Explanation: Error. A parity error occurred on the port.

User Action: Check parity of the device and make sure it matches of the host. (LAT and TELNET parity is set in the terminal server.) If parity is set correctly, then verify that all cabling is grounded and terminated properly.

portsyncerror, Unable to change file descriptor to blocking,

Explanation: Error. An error occurred when using ioctl( ) to place a file descriptor in blocking mode.

User Action: Examine the log file (BASEstar Open) or stderr to determine the error and then take corrective action based on the error found.

portsyncstatus, Unable to change file descriptor to blocking for port XX, errno = X,

Explanation: Error. An error occurred when using ioctl( ) to place a file descriptor in blocking mode.

User Action: Look up the error and then take corrective action based on the error found.

portsyntax, Improper syntax for intenet address,

Explanation: Error. There is an error in the syntax of the port address.

User Action: Check the syntax of the address and correct it to be a well formed internet address.

recverror, Read (recv) error occurred on the port,

Explanation: Error. An error was found when reading from the port. Examine the log file (BASEstar Open) or stderr to determine the errno returned by the read.

User Action: Examine the log file (BASEstar Open) or stderr to determine the errno returned by the recv call. Then take action based on the error found.

recvstatus, Read (recv) error on port XX, errno = X,

Explanation: Error. An error was found when reading from the port. The error number for the error is given in the message.

User Action: Look up the error and take action based on the error found.

socketerror, Error in obtaining a socket for port,

Explanation: Error. An error was found when issuing a 'socket' call to obtain a socket for a TCP/IP connection.

User Action: Examine the log file (BASEstar Open) or stderr to determine the errno returned by the socket call. Then take action based on the error found.

socketstatus, Error in obtaining a socket for port XX, errno = X,

Explanation: Error. An error was found when issuing a 'socket' call to obtain a socket for a TCP/IP connection.

User Action: Look up the error and take action based on the error found.

telnetiac, TELNET command XX received on port XX,

Explanation: Error. A TELNET command was received when in binary mode. In binary mode no TELNET commands are expected.

User Action: No user action should be necessary. The DAS will abort the connection and reset binary mode when it reconnects.

ttbreakerror, An error occurred when sending a break to a port,

Explanation: Error. An error was returned when sending a break to the port.

User Action: Examine the log file (BASEstar Open) or stderr to determine the error that occurred. Then take action based on the error found.

ttcharerror, An error occurred when defining TT characteristics,

Explanation: Error. An error occurred when setting up TT characteristics. The TT characteristics are set when a connection is initiated with the VMD.

User Action: An error message detailing the error should have been logged. Examine the error to determine the proper corrective action.

ucxseterror, Error returned from SYS$QIO - IO$_SETMODE,

Explanation: Error. An error occurred when attempting to set the socket address. This UCX call is equivalent to the C Socket socket ( ).

User Action: Examine the log file (BASEstar Open) or stderr to determine the error that occurred. Then take corrective action based on the error that was found.

ucxsetstatus, Status for port XX returned from SYS$QIO - IO$_SETMODE, XX,

Explanation: Error. An error occurred when attempting to set the socket address. This UCX call is equivalent to the C Socket socket ( ).

User Action: Examine the error status and look up the error to determine the cause and corrective action to take.

unexpectiac, An unexpected "IAC" was received while in TELNET binary mode,

Explanation: Error. An "IAC" (Is A command) character was received after a TELNET line was negotiated to transmit and receive in binary mode. In binary mode, the only expected TELNET control command is the IAC character itself.

User Action: No action should be necessary. The DAS will abort the connection if this should occur and then attempt to renegotiate binary transmission/reception when reconnecting.

vmsconnectstatus, Status for port XX returned from SYS$QIO - IO$_ACCESS, XX,

Explanation: Error. An error occurred when attempting to connect to an Internet address. This UCX call is equivalent to the C connect ( ).

User Action: Look up the error status to determine the error that occurred. Then take corrective action based on the error that was found.

vmsreaderror, QIO read error on port,

Explanation: Error. An error was found when reading from an OpenVMS port. Examine the log file (BASEstar Open) or stderr to determine the status returned by the read.

User Action: Look up the error and take action based on the error found.

vmsreadstatus, QIO read error on port XX, XX,

Explanation: Error. An error occurred when reading data from an OpenVMS port.

User Action: Examine the error message and take corrective action based on the error message.

vmsrecverror, Error returned from SYS$QIO - IO$_READVBLK,

Explanation: Error. An error was found when reading from the port. This UCX call is equivalent to the C Socket recv ( ).

User Action: Examine the log file (BASEstar Open) or stderr to determine the error that occurred. Then take action based on the error found.

vmsrecvstatus, Status for port XX:X returned from SYS$QIO - IO$_READVBLK, XX,

Explanation: Error. An error was found when reading from the port. This UCX call is equivalent to the C Socket recv ( ).

User Action: Look up the error status to determine the error that occurred. Then take corrective action based on the error that was found.

vmsttbreakstatus, Status for port XX returned from SYS$QIO - IO$_SETMODE, XX,

Explanation: Error. An error was returned when sending a break to the port.

User Action: Look up the error status to determine the error that occurred. Then take corrective action based on the error that was found.

vmsttcharstatus, Status for port XX returned from SYS$QIOW (IO$_SETMODE), XX,

Explanation: Error. An error occurred when setting port characteristics.

User Action: Look up the error number to determine the cause of the error and take appropriate action based on the error code.

wont, TELNET server refused to accept TELNET option,

Explanation: Error. The TELNET server responded with "WON'T" to a request, "DO", for the requested TELNET option.

User Action: Enable the apropriate option on the TELNET server or substitute a server that supports the requested option.

wontdobin, TELNET server refused to send binary data,

Explanation: Error. The TELNET server responded with "WON'T" to a request, "DO", for the server to send binary data (TELNET option RFC 856).

User Action: Enable binary mode transmission on the TELNET server or substitute a server that supports binary mode transmission/reception.

wrongdevice, An invalid device type was detected for a port,

Explanation: Error. A device type other than a terminal was detected for the indicated port.

User Action: Modify the ODS port definition to use a valid TT type device.

B

DAS Error Messages

This appendix contains returned values and associated error messages specific to the DAS for Modicon programmable controllers.

aborting, The connection to the device is being aborted,

Explanation: Error. The connection to the device is being aborted by an error or by user request. No more requests will be processed by the DAS until the connection is restored.

User Action: If the abort was caused by a device error, take action based on the error that occurred. If the abort was a result of a user request. Reconnect to the device before attempting further operations.

addresstoolong, Variable address string too long,

Explanation: Error. A variable address that is being defined is too long. An address must be 30 characters of less in length.

User Action: Redefine the variable using a shorter address.

allocerror, An error occurred when allocating a VMS port,

Explanation: Error. An error occurred when allocating a VMS port. The allocation is done when a connection is initiated with the VMD.

User Action: An error message detailing the error should have been logged. Examine the error to determine the proper corrective action.

allocplusstatus, Status for line XX and path X returned from SYS$QIOW | IO$_ ALLOC, XX,

Explanation: Error. An error occurred when allocating a Modbus Plus path. If the path number is greater than 0 then at least one path was allocated to process Modbus Plus messages.

User Action: Examine the error message to determine the cause of the error and take appropriate action based on the error message.

apptypeunsupp, The given application type is not supported for this DAS,

Explanation: Error. The application type that was specified in the variable definition is not supported by this DAS.

User Action: Change the application type to a valid type for this DAS.

assignerror, An error occurred when assigning a channel to a VMS port,

Explanation: Error. An error occurred when assigning a channel to a VMS port. The assignment is done when a connection is initiated with the VMD.

User Action: An error message detailing the error should have been logged. Examine the error to determine the proper corrective action.

assignplusstatus, Status for line XX and path X returned from SYS$ASSIGN, XX,

Explanation: Error. An error occurred when assigning a channel to a Modbus Plus path.

User Action: Examine the error message to determine the cause of the error and take appropriate action based on the error message.

badaddress, VMD XX receieved data for address X, but this address does not exist on the host,

Explanation: Error. A variable was received from a programmable controller and the address did not match any variables defined on the host VMD.

User Action: Define a variable with the correct address on the host VMD or change the address of the variable being sent from the programmable controller.

badroutingbytes, Routing byte X = !UW is not a valid value on line XX,

Explanation: Error. The host received an unsolicted write message from the programmable controller, but the first five words of the programmable controller did not contain valid node routing information for the programmable controller sending the data.

User Action: Modify the first five words of the programmable controller data area to send valid routing information for the programmable controller sending the data.

datatoosmall, Programmable controller data too small for the data size of the variable,

Explanation: Error. A variable was received from a programmable controller and the address matched a host variable, but the size of the data defined for the variable is larger than the size of the data actually received.

User Action: Define a variable with the correct size as the data being sent or change the size of the data being sent by the programmable controller.

devfailed, Failure in associated device,

Explanation: Error. The slave's programmable controller has failed to respond to a message or an abortive error occurred.

User Action: An irrecoverable error in the programmable controller occurred. Replace or repair the defective programmable controller.

devilladdress, Illegal data address,

Explanation: Error. The address referenced in the data field is not an allowable address for the addressed slave location.

User Action: Check the model of the programmable controller and the documentation to determine a valid address for the programmable controller. Then modify the address to a valid address for this programmable controller.

devillegalfnc, Illegal function,

Explanation: Error. The message function received is not an allowable action for the addressed slave. This error is most likely caused by setting an incorrect application profile, so a command is being sent to a wrong model programmable controller.

User Action: Check that the corrrect application profile is being used for this model of programmable controller.

devillvalue, Illegal data value,

Explanation: Error. The value referenced in the data field is not an allowable value for this addressed slave location.

User Action: Check the model of the programmable controller and the documentation to determine a valid value for the programmable controller. Then modify the value to a valid value for this programmable controller.

internalerror, Internal error,

Explanation: Error. An internal error occurred in the DAS.

User Action: Submit an SPR describing the circumstances that caused the error.

invaddresstype, Invalid address type,

Explanation: Error. A variable that is being defined is defined as an illegal address type. The unconstrained address type is used for Modicon 584 and 984 extended files. The numeric address type is used for all other registers/coils.

User Action: Redefine the variable using the proper unconstrained or numeric address type.

invalidaddress, The specified programmable controller address is not supported by this DAS,

Explanation: Error. An address was sent by a programmable controller but the address is not supported by this DAS.

User Action: Reprogram the programmable controller and specify a valid address for this DAS. Consult the DAS documentation to determine a valid address.

invaliddatamaster, Invalid value for number of data master paths,

Explanation: Error. The number of data master paths must be between 1-7. I/O can occur on all paths simultaneously, so the more paths, the greater the theoretical throughput. However, if all paths are used by the DAS, Modicon utilities such as NDU cannot be run while the DAS is running.

User Action: Use ODS to enter a number for the number of data master paths between 1-7.

invaliddataslave, Invalid value for number of data slave paths,

Explanation: Error. The number of data slave paths must be between 0-7. I/O can occur on all paths simultaneously, so the more paths, the greater the theoretical throughput. The programmable controller program must code the path as part of the MSTR instruction. If the number of paths is set to 0, then

no slave paths are allocated, so no data can be sent from the programmable controller(s).

User Action: Use ODS to enter a number for the number of data slave paths between 0-7.

invaliddrop, Invalid drop time specified in MODBUS_TELNET ODS entry,

Explanation: Error. The drop time specified is not a valid number. The drop time must be a number greater than 0.

User Action: Use ODS to enter a valid drop time attribute.

invalidfunction, Invalid function code received in a message from the programmable controller,

Explanation: Error. A message was received from the programmable controller, but the function code used by the programmable controller in the write command is not supported by the DAS.

User Action: Reprogram the programmable controller to send the correct function code.

invalidgroup, Variables in a group must all belong to the same address type,

Explanation: Error. A group read was attempted without all the variables in the group belonging to the same address reference type (coil, holding register, input, etc.).

User Action: Examine the members of the group to determine which variable does not belong, remove that variable from the list and retry the group read.

invalidgroupmem, Variable XX in a group read for device XX does not belong,

Explanation: Error. A group read was attempted without all the variables in the group belonging to the same address reference type (coil, holding register, input, etc.).

User Action: Remove the incorrect variable from the list and retry the group read.

invalidindpath, Invalid indication path flag attribute specified in the MODPLUS_LINE ODS entry,

Explanation: Error. The routing path flag attribute must be one of TRUE, ON, YES, 1 or FALSE, OFF, NO, 0.

User Action: Use ODS to enter a valid routing path flag attribute.

invalidipaddress, Invalid IP_ADDRESS ODS attribute for the given line,

Explanation: Error. The IP_ADDRESS must be less than 50 characters in length.

User Action: Enter an IP address that is 50 characters or less in length.

invalidmodem, Invalid modem attribute specified in the MODBUS_LINE ODS entry,

Explanation: Error. The modem attribute must be one of TRUE, ON, YES, 1 or FALSE, OFF, NO, 0.

User Action: Use ODS to enter a valid modem attribute.

invalidnodeaddress, Invalid node address number in the MODPLUS_NODE_ PATH ODS entry,

Explanation: Error. Modbus node address numbers must be a number between 1-64. The node address is a DIP switch setting on your Modbus Plus card.

User Action: Use ODS to enter a node address number between 1-64. Make sure the address matches the DIP switch setting on the Modbus Plus card.

invalidparity, Invalid parity specified in the MODBUS_LINE ODS entry,

Explanation: Error. The parity defined must be either EVEN, ODD or NONE.

User Action: Use ODS to enter a valid parity (EVEN, ODD, NONE).

invalidport, Invalid port name specified in the ODS entry,

Explanation: Error. Port names must be 10 characters or less.

User Action: Use ODS to enter a port name that is less than 10 characters.

invalidportnum, Invalid PORT_NUMBER ODS attribute for the given line,

Explanation: Error. The PORT_NUMBER must be less than 10 characters in length.

User Action: Enter a PORT_NUMBER address that is 10 characters or less in length.

invalidprobe, Invalid probe time specified in MODBUS_TELNET ODS entry,

Explanation: Error. The probe time specified is not a valid number. The probe time must be a number greater than 0.

User Action: Use ODS to enter a valid probe time attribute.

invalidretries, Invalid retries number in the ODS entry,

Explanation: Error. The retries attribute must be between 0-20.

User Action: Use ODS to enter a retries number between 0-20.

invalidslaveaddress, Invalid slave address number in the MODBUS_PATH ODS entry,

Explanation: Error. Modbus slave address numbers must be a number between 1-247.

User Action: Use ODS to enter a slave address number between 1-247.

invalidslavepath, Invalid slave routing path in the MODPLUS_SLAVE_PATH ODS entry,

Explanation: Error. Modbus Plus routing paths are of the form xx.xx.xx.xx.xx. Nodes must be in the range 1-64. If using a multiplexor bridge to a networked Modbus slave device the last routing byte can be 1-247. All five bytes must be specified. Unused routing bytes must be 0.

User Action: Use ODS to enter a slave routing path with the proper syntax and valid node/slave address numbers.

invalidspeed, Invalid speed (baud rate) specified in the MODBUS_LINE ODS entry,

Explanation: Error. Baud rates from 110 to 19200 are supported for the Modicon DAS.

User Action: Use ODS to enter a speed from 110 to 19200.

invalidtimeout, Invalid timeout value in the ODS entry,

Explanation: Error. An invalid timeout value was entered for the timeout value in ODS. Timeouts must be between 2 and 10 seconds.

User Action: Use ODS to enter a timeout value between 2-10.

invalidtrace, Invalid trace attribute specified in the MODBUS_LINE or MODBUS_PATH ODS entry,

Explanation: Error. The trace attribute must be one of TRUE, ON, YES, 1 or FALSE, OFF, NO, 0.

User Action: Use ODS to enter a valid trace attribute.

invidnotfound, Matching invocation ID not found for VMD = XX, ID = !UW,

Explanation: Error. A response was received from the device, but no matching request was found either because the request timed out or because this is a duplicate response. A request and response have a unique invocation ID (transaction number) that allows the request and response to be matched up.

User Action: Increase the device timeout or analyze the network to determine where network delays could be ocurring and then take action to decrease network delays.

locvmdnotunique, Local VMD not unique,

Explanation: Error. More than one local VMD has been used when connecting to a Modbus Plus network and the line parameter flag specifying that the path is encoded in the write indication data has not been set.

User Action: Use only one local VMD or change the ODS line parameter IND_ROUTE_PATH to indicate that a path is encoded in data sent from the programmable controller.

ltnthreadexcep, Listen thread exception, Status = X,

Explanation: Error. The listen thread (listening for connection requests from TCP/IP clients) has received an unexpected exception (error).

User Action: Examine the Status for the thread and take based on the error found.

maxlines, The maximum number of lines have already been configured,

Explanation: Error. The maximum number of lines have already been for this system. A maximum of four Modbus Plus cards can be configured for one VAX.

User Action: Reconfigure the /LINE_NAME parameter to match a valid configuration for this system.

mmsaddrtypeunsupp, The given MMS type is unsupported for the given address,

Explanation: Error. The MMS type is not supported for the variable address given.

User Action: Modify the MMS type to one supported for the given address or modify the address to one supported for the MMS type.

mmstypeunsupp, The given MMS type is not supported for this DAS,

Explanation: Error. The MMS type that was specified in the variable definition is not supported by by this DAS.

User Action: Change the MMS type to a valid type for this DAS.

modbuserror, A modbus error occurred,

Explanation: Error. A Modbus error occurred when reading or writing to the programmable controller.

User Action: Retry issuing the command.

neterr, Modbus network error occurred,

Explanation: Error. A network error on the Modicon Modbus occurred.

User Action: Examine the network status in the returned IOSB to determine the specific network error that occurred and take action based on that error.

noaddress, No "NODE_ADDRESS" or "SLAVE_ROUTING" ODS entry for the given application simple name,

Explanation: Error. The Modicon DAS requires a NODE_ADDRESS or SLAVE_ROUTING attribute defined for the ODS entry corresponding to the application simple name entry for the VMD. The NODE_ADDRESS attribute is defined for MODPLUS_NODE_PATH object class within ODS. The SLAVE_ROUTING attribute is defined for the MODPLUS_SLAVE_PATH object class within ODS.

User Action: Use the MODPLUS_NODE_PATH or MODPLUS_SLAVE_ PATH object class within ODS to define the application simple name for this VMD. Use a valid Modbus Plus node address number if defining the NODE_ADDRESS attribute. Use a valid Modbus Plus slave routing path if defining the SLAVE_ROUTING attribute.

noipaddress, No "IP_ADDRESS" ODS entry for the given line,

Explanation: Error. The DAS requires an IP_ADDRESS attribute defined in ODS. The IP_ADDRESS attribute is defined for the MODBUS_TELNET class within ODS.

User Action: Use the MODBUS_TELNET object class within ODS to define the IP_ADDRESS attribute for this line. Use a valid IP address for this attribute.

nolicense, No software license.,

Explanation: Error. The Modicon DAS requires a software license.

User Action: Install the license PAK for the Modicon DAS.

noline, No "/LINE=" ODS entry for the given application simple name,

Explanation: Error. The Modicon DAS requires a /LINE= attribute defined for the ODS entry corresponding to the application simple name entry for the remote VMD. The /LINE= attribute is defined for MODBUS_PATH and MODPLUS_SLAVE_PATH object classes within ODS.

User Action: Use the MODBUS_PATH object class (for Modbus) or MODPLUS_SLAVE_PATH object class (for Modbus Plus) within ODS to define the application simple name for this VMD. Use a valid line name when defining the /LINE= attribute.

nomodem, No "MODEM" ODS entry for the given line,

Explanation: Error. The Modicon DAS requires a MODEM attribute defined in ODS. The MODEM attribute is defined for the MODBUS_LINE object class within ODS.

User Action: Use the MODBUS_LINE object class within ODS to define the MODEM attribute for this line. Modem control is either enabled (TRUE, ON, YES, 1) or disabled (FALSE, OFF, NO, 0).

noparity, No "PARITY" ODS entry for the given line,

Explanation: Error. The Modicon DAS requires a PARITY attribute defined in ODS. The PARITY attribute is defined for the MODBUS_LINE object class within ODS.

User Action: Use the MODBUS_LINE object class within ODS to define the PARITY attribute for this line. Use a valid parity (EVEN, NONE) when defining the PARITY attribute.

noport, No "PORT" ODS entry for the given line,

Explanation: Error. The Modicon DAS requires a PORT attribute defined in ODS. The PORT attribute is defined for the MODBUS_LINE object class within ODS.

User Action: Use the MODBUS_LINE object class within ODS to define the PORT attribute for this line. Use a valid port name when defining the PORT attribute.

noportnumber, No "PORT_NUMBER" ODS entry for the given line,

Explanation: Error. The DAS requires a PORT_NUMBER attribute defined in ODS. The PORT_NUMBER attribute is defined for the MODBUS_TELNET class within ODS.

User Action: Use the MODBUS_TELNET object class within ODS to define the PORT_NUMBER attribute for this line. Use a valid IP port number for this attribute.

noretries, No "RETRIES" ODS entry for the given line,

Explanation: Error. The Modicon DAS requires a RETRIES attribute defined in ODS. The RETRIES attribute is defined for the MODBUS_LINE object class within ODS.

User Action: Use the MODBUS_LINE object class within ODS to define the RETRIES attribute for this line. Use a valid number when defining the RETRIES attribute.

noslaveaddress, No "SLAVE_ADDRESS" ODS entry for the given application simple name,

Explanation: Error. The Modicon DAS requires a SLAVE_ADDRESS attribute defined for the ODS entry corresponding to the application simple name entry for the remote VMD. The SLAVE_ADDRESS attribute is defined for MODBUS_PATH object class within ODS.

User Action: Use the MODBUS_PATH object class within ODS to define the application simple name for this VMD. Use a valid Modbus slave address number when defining the SLAVE_ADDRESS attribute.

nospeed, No "SPEED" ODS entry for the given line,

Explanation: Error. The Modicon DAS requires a SPEED attribute defined in ODS. The SPEED attribute is defined for the MODBUS_LINE object class within ODS.

User Action: Use the DH_LINE object class within ODS to define the SPEED attribute for this line. Use a valid baud rate when defining the SPEED attribute.

notimeout, No "TIMEOUT" ODS entry for the given line,

Explanation: Error. The Modicon DAS requires a TIMEOUT attribute defined in ODS. The TIMEOUT attribute is defined for the MODBUS_LINE object class within ODS.

User Action: Use the MODBUS_LINE object class within ODS to define the TIMEOUT attribute for this line. Use a valid number when defining the TIMEOUT attribute.

notrace, No "TRACE" ODS entry,

Explanation: Error. The Modicon DAS requires a TRACE attribute defined in ODS. The TRACE attribute is defined for the MODBUS_PATH, MODBUS_LINE, MODPLUS_LINE, MODPLUS_SLAVE_PATH and MODPLUS_NODE_PATH object classes within ODS.

User Action: Use the appropriate object class within ODS to define the TRACE attribute for this line. Trace is either enabled (TRUE, ON, YES, 1) or disabled (FALSE, OFF, NO, 0).

parsefailed, The parsing of a variable address failed,

Explanation: Error. When defining a variable, the parsing of the address failed. Either the syntax or the value of the address is incorrect.

User Action: Consult the user documentation to determine a valid address for the variable being defined and redefine the variable using the correct syntax.

rcvthreadexcep, Receive thread exception for device XX, Status = X,

Explanation: Error. The receiver thread (waiting for input in response to a request for data) has received an unexpected exception (error).

User Action: Examine the Status for the thread and take based on the error found.

reqpending, Conclude rejected because of a pending read/write request,

Explanation: Error. The conclude request was rejected because either a read or a write request was currently pending.

User Action: Either abort the connection or wait for the current request(s) to complete before retrying the conclude.

retriesexceeded, The retry count was exceeded,

Explanation: Error. When trying to communicate to the device, the retry count was exceeded. The retry being exceeded means that the message being sent to the device was garbled during transmission and did not match the checksum.

User Action: Check cabling to ensure there are no shorts or breaks and is properly grounded. Make sure the cable is routed such that it does not pass close to electrical interference such as transformers and high power cables.

smalldata, VMD XX receieved data for address X, but the data was too small,

Explanation: Error. A variable was received from a programmable controller and the address matched a host variable, but the size of the data defined for the variable is larger than the size of the data actually received.

User Action: Define a variable with the correct size as the data being sent or change the size of the data being sent by the programmable controller.

solnotsupported, Solicited operations (read/write) are not supported for this VMD,

Explanation: Error. Solicited operations are not supported for this VMD. This VMD is used to receive indications (info reports) from the VMD and is not available to perform read/write operations. The programmable controller did not respond when attempting to read a response from the programmable controller. The response could be part of a read or a write operation.

User Action: Check all cabling to make sure that it is properly shielded, grounded and seated. Verify that the communication parameters are set up properly on both programmable controller and the host. Verify that the correct slave address has been selected on the programmable controller and on the host.

timeout, Modbus timeout when reading from the programmable controller,

Explanation: Error. The programmable controller did not respond when attempting to read a response from the programmable controller. The response could be part of a read or a write operation.

User Action: Check all cabling to make sure that it is properly shielded, grounded and seated. Verify that the communication parameters are set up properly on both programmable controller and the host. Verify that the correct slave address has been selected on the programmable controller and on the host.

tmrthreadexcep, Timer thread exception for device XX, Status = X,

Explanation: Error. The timer thread (timing out responses to requests for data) has received an unexpected exception (error).

User Action: Examine the Status for the thread and take based on the error found.

twofewbytes, Not enough bytes sent in unsolicited write modbus packet,

Explanation: Error. The host received an unsolicted write message from the programmable controller, but not enough bytes were received to decode the sending programmable controller's routing address. When configured using ODS, the DAS expects the first five words of the data to be the address of the programmable controller sending the data. This error is generated if ten or less words are sent by the programmable controller.

User Action: Modify the programmable controller program to send more than five words of data (with the first five being the address of the programmable controller) or use ODS to to tell the DAS not to expect routing information to be contained in the data.

unexpfunction, Unexpected function received from programmable controller XX. The received function code was 'XX',

Explanation: Error. An unexpected function code was received in a message from the programmable controller.

User Action: Reprogram the programmable controller to send the correct function code.

unsupported, The requested function is not supported by the DAS,

Explanation: Error. An attempt was made to execute a OMNI function that is not supported by this DAS.

User Action: Consult the user documentation to determine which functions are supported by the DAS and limit the OMNI functions used to those supported by the DAS.

variablenotfound, Variable not found with the specified address,

Explanation: Error. A variable was received from a programmable controller and the address did not match any variables defined on the host.

User Action: Define a variable with the correct address on the host or change the address of the variable being sent from the programmable controller.

vmsioerror, An error occurred when doing I/O to a Modbus Plus channel,

Explanation: Error. An error occurred when reading or writing to a Modbus Plus channel. The channel is assigned to a path used to communicate with the Modbus Plus line.

User Action: An error message detailing the error should have been logged. Examine the error to determine the proper corrective action.

vmsiostatus, I/O error on Modbus Plus line XX, QIO status, XX,

Explanation: Error. An error occurred when reading/writing data on a Modbus Plus line.

User Action: Examine the error message to determine the cause of the error and take appropriate action based on the error message.

writenotsupport, Writes are not supported for the specified address,

Explanation: Error. An attempt was made to write to an address that the Modbus protocol does not allow write access to.

User Action: Consult the documentation to determine addresses that can be written to and change the address of the variable or only perform reads when using this variable.

wrongindication, An indication from XX was received on line XX,

Explanation: Error. A write inidication was received but a connection has not established to receive indications on that line for that programmable controller. The routing bytes in the message are obtained from the first five words of the unsolicted message.

User Action: Establish a connection to receive indications on the line. One VMD on the line is dedicated to receive indications for that line.

Index

A

Abort function, 3–2
Abort indication, 3–11
Accessing DAS functions, 3–1
Additional software
  Modbus Plus, 2–4
  Modbus TCP/IP, 2–4
Address syntax, 3–11
  Modicon 184, 3–12
  Modicon 384, 3–12
  Modicon 484, 3–12
  Modicon 584, 3–13
  Modicon 884, 3–12
  Modicon 984, 3–13
  Modicon Micro 84, 3–11

C

CLI
  accessing, 3–1
Conclude function, 3–2
Configuration
  editing configuration files, 2–9
  ODS registrations, 2–10
  protocol profile, 2–13
  VMD definitions, 2–13
Configuration files, 2–15
Configuring ports, 2–18
Configuring TELNET, 2–18
Connect function, 3–2

D

DAS
  See Device access software
Data types, 3–11
Device access software
  accessing functions, 3–1
  address syntax, 3–11
  description of, 1–1
  functions, 1–2
  installation of, 2–1
  installation requirements, 2–1
  overview, 1–1
  supported client functions, 3–2
  supported data types, 3–11
Device access software (cont'd)
  supported devices, 1–2
  supported functions, 1–2
  supported server functions, 3–2
  using, 3–1
Device communications
  cabling, 2–1
  overview, 1–1
  tracing, 2–19
Devices
  Programmable controllers supported, 1–3

F

Files, created during installation, 2–7
Functions, 3–1
  abort, 3–2
  conclude, 3–2
  connect, 3–2
  read data point, 3–5
  read variable, 3–2
  supported, 3–1
  write data point, 3–5
  write variable, 3–2

I

Indications
  abort, 3–11
  receiving, 3–7
  write, 3–7
Installation, 2–1
  files created, 2–7
  messages, 2–7
  procedure, 2–5
Installation requirements, 2–1
  additional software, 2–4
  disk space, 2–5
  hardware, 2–1
  software, 2–4

M

Messages
  DAS error and return values, B–1
  I/O error and return values, A–1
  installation, 2–7

MMS data type, 3–11
Modicon 184
address syntax, 3–12
supported data types, 3–12
Modicon 384
address syntax, 3–12
supported data types, 3–12
Modicon 484
address syntax, 3–12
supported data types, 3–12
Modicon 584
address syntax, 3–13
supported data types, 3–13
Modicon 884
address syntax, 3–12
supported data types, 3–12
Modicon 984
address syntax, 3–13
supported data types, 3–13
Modicon Micro 84
address syntax, 3–11
supported data types, 3–11

O

ODS registration
Example, 2–10
Modbus, 2–10
Modbus Plus, 2–11
Modbus TCP/IP, 2–12
ODS registrations, 2–10

P

Plant-floor equipment, setting up, 2–19
Postinstallation tasks, 2–9
Problem reporting, 2–20
Product failure, 2–20
Protocol profile
184, 2–13
384, 2–13
484, 2–13
584, 2–13
884, 2–13
984, 2–13
micro 84, 2–13
Modbus Plus, 2–13
Modbus TCP/IP, 2–13
Quantum Automation Series, 2–13

Q

Quantum Automation Series
protocol profile, 2–13

R

Read data point function, 3–5
Read variable function, 3–2

S

Supported data types, 3–11
Modicon 184, 3–12
Modicon 384, 3–12
Modicon 484, 3–12
Modicon 584, 3–13
Modicon 884, 3–12
Modicon 984, 3–13
Modicon Micro 84, 3–11

T

TELNET, 2–18
binary transfer, 2–18
Trace
ODS configuration, 2–10
using, 2–19

V

VMD definitions, 2–13

W

Write data point function, 3–5
Write indication, 3–7
Write variable function, 3–2