BASEstar™ Classic DAS for Modicon® Programmable Controllers

Installation and User's Guide

Order Number: AA-R216B-TE

April 2000

This manual describes how to install and use the DAS for Modicon Programmable Controllers for BASEstar Classic on OpenVMS.

Revision/Update Information:	This is a revised document.
Operating System and Version:	OpenVMS/Alpha Version 6.1
Operating System and Version:	OpenVMS/VAX Version 5.5-2
Interface Software and Version:	BASEstar Classic Version 3.4
Software Version:	BASEstar Classic DAS for Modicon Programmable Controllers, Version 3.4A

Compaq Computer Corporation
Houston, Texas

April 2000

© 1990 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.

584, 984, 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 2–2
2.1.3 Additional Software 2–2
2.1.4 Disk Space 2–2
2.2 Installation Procedure 2–2
2.2.1 Files Created During Installation 2–4
2.2.2 Installation Messages 2–5
2.3 Postinstallation Tasks 2–6
2.3.1 Editing the Configuration File 2–6
2.3.1.1 Editing Type Records 2–6
2.3.1.2 Editing Path Records 2–6
2.3.1.3 Editing Device Records 2–9
2.3.1.4 Network Address Using Modbus 2–9
2.3.1.5 Network Address Using Modbus Plus 2–9
2.3.1.6 Network Address for the Modbus Plus Host 2–9
2.3.1.7 Network Address Using TCP/IP Ethernet Interface Module 2–10
2.3.2 Setting the SPT Block Parameter 2–10
2.3.3 Configuring Ports 2–11
2.3.4 Setting DAS-Specific Parameters 2–12
2.3.4.1 Setting Modbus Parameters 2–12
2.3.4.2 Setting Unsolicited Parameters 2–13
2.3.4.3 Setting TCP/IP Parameters 2–13
2.3.5 Setting Up Plant-Floor Equipment 2–14
2.4 Tracing Device Communications 2–14
2.5 Failures During Product Use 2–15

3 Using the DAS

3.1 Accessing DAS Functions 3–1
3.2 Structured Data 3–1
3.3 Supported Functions 3–2
3.3.1 Start and Stop 3–2
3.3.2 Upload and Download 3–2
3.3.3 Read and Write Data Functions 3–3
3.3.3.1 Quantum, Modicon 984 and Modicon 584 3–4
3.3.4 Read Status 3–5
3.3.4.1 Unsolicited Counters 3–6
3.4 Automatic Data Collection 3–6
3.4.1 Polled Data Collection 3–6
3.4.2 Unsolicited Data Collection 3–6
3.4.2.1 Unsolicited ID 3–7

A Equipment Configuration

B Logged Messages

B.1 Modbus NI Logged Messages B–1
B.2 Modbus Plus NI Logged Messages B–2
B.3 Modbus TCP/IP NI Logged Messages B–4
B.4 PE Logged Messages B–6

Index

Examples

3–1 Read Data Screen 3–3
3–2 Read Data Screen with Structured Data 3–3
3–3 Write Data Screen 3–4
3–4 Write Data Screen with Structured Data 3–4
3–5 Read Status Screen 3–5

Figures

1–1 DAS Communications 1–2
A–1 Processor-to-Modem Connection A–1
A–2 Modbus Block Diagram A–2
A–3 Modbus Plus Block Diagram A–3
A–4 Modbus TCP/IP Block Diagram A–4

Tables

1–1	Modicon Programmable Controllers	1–3
2–1	Disk Space Requirements	2–2
2–2	Files Created During Installation	2–4
2–3	Files Used by Simple Compare	2–4
2–4	Modbus Path Parameters	2–7
2–5	Modbus Line Parameters	2–7
2–6	Modbus-Plus Path Parameters	2–8
2–7	Modbus TCP/IP Path Parameters	2–8
2–8	SPT Static Block Sizes	2–10
2–9	SPT Dynamic Block Sizes	2–11
2–10	Modbus Parameters	2–12
2–11	Unsolicited Parameters	2–13
2–12	TCP/IP Parameters	2–13
3–1	Modicon Devices and Functions	3–2

Preface

This document describes how to install and use the BASEstar Classic DAS for Modicon programmable controllers.

Intended Audience

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

- BASEstar Classic for OpenVMS software
- BASEstar Classic DAS for Modicon programmable controllers

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

Readers of this document should have a solid understanding of OpenVMS operations and administration, as well as OpenVMS application software. In addition, knowledge of Modicon programmable controllers and the specific requirements of the installation site is essential.

Document Structure

This document is organized as follows:

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

Associated Documents

Further information on BASEstar Classic can be found in the following documents:

- *BASEstar Classic Installation Guide*
- *BASEstar Classic Configuration and Tuning Guide*
- *BASEstar Classic Menu Interface User's Guide*
- *BASEstar Classic Command Line Interface User's Guide*
- *BASEstar Classic Introduction to Callable Services*
- *BASEstar Classic Guide to Writing Device Access Software*
- *BASEstar Classic Application Programming Interface Reference Guide*

Further information on Digital TCP/IP Services for OpenVMS can be found in the following documents:

- *Digital TCP/IP Services for OpenVMS Installation and Configuration*
- *Digital TCP/IP Services for OpenVMS User's Guide*
- *Digital TCP/IP Services for OpenVMS Management*
- *Digital TCP/IP Services for OpenVMS Management*
- *Digital TCP/IP Services for OpenVMS Management Command Reference*
- *Digital TCP/IP Services for OpenVMS System Services and C Socket Programming*

Further information on Modicon communications can be found in the following documents:

- *Modbus Protocol Reference Guide*
- *Modbus System Planning Guide*
- *J478 Stand-Alone Modem Guide*
- *Modicon DEC Host Based Devices User's Guide*
- *Modicon Quantum Ethernet TCP/IP Module User 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 Classic DAS for Modicon programmable controllers. It also briefly describes Modicon programmable controllers and the supported functions for the DAS.

1.1 Description

The DAS for Modicon programmable controllers allows you to access Modicon programmable controllers through BASEstar Classic device connection management capabilities. Device connection management is the device connection/control component of BASEstar Classic for OpenVMS. BASEstar Classic for OpenVMS is software designed to facilitate the integration of manufacturing equipment and applications, accelerate the development of integrated manufacturing systems, and provide an architecture for consistent development of manufacturing applications.

Using the DAS for Modicon programmable controllers, users or applications can perform a variety of device access functions which include: reading and writing data, uploading and downloading, and reading the status of a device.

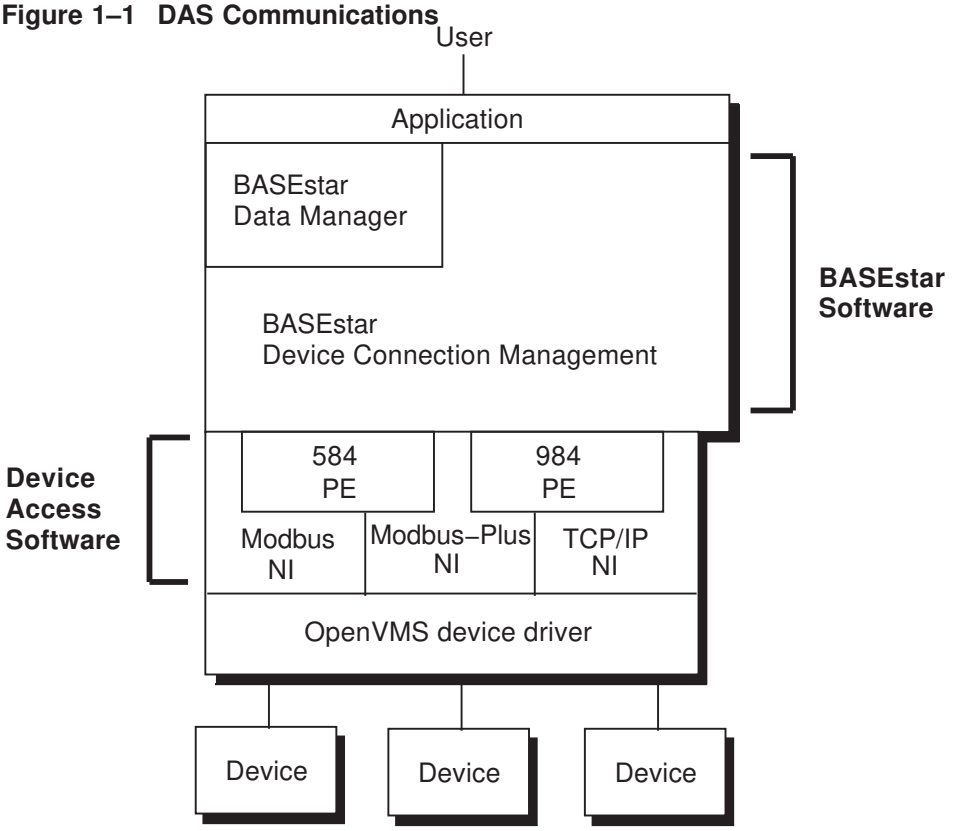
1.2 Device Communications

The DAS for Modicon programmable controllers consists of two protocol emulators (PE) and three network interfaces (NI). A PE and NI work together to provide device-specific communications for BASEstar Classic device connection management's generic callable services.

The PE translates BASEstar Classic device connection management's generic services into a device-understandable format. The PE also converts device-specific protocol into a BASEstar Classic device connection management format. The NI works directly with a OpenVMS driver to send data to and receive data from plant-floor devices. The NI communicates the requests and data translated by the PE. Figure 1–1 shows how the DAS facilitates communications between BASEstar Classic device connection management and the devices.

The Modicon 584 PE is supported only for use with the Modbus NI. The Modicon 984 PE can be used with any of the supplied NIs.

Figure 1–1 DAS Communications



1.3 Supported Functions and Devices

You can perform only the BASEstar Classic device connection management functions that are supported by a device's PE. These functions can be accessed through BASEstar Classic device connection management's menu system, commands, and callable services.

The DAS for Modicon programmable controllers supports the following BASEstar Classic device connection management functions:

- Start and stop operations on a device
- Upload the contents of a device's memory to an OpenVMS file
- Download an OpenVMS file to a device's memory
- Read data from a specific address in a device's memory
- Write data to a specific address in a device's memory
- Read status of a device

The DAS for Modicon programmable controllers supports a variety of programmable controllers, as shown in Table 1–1.

Table 1–1 Modicon Programmable Controllers

Device	Upload	Download	Start	Stop	Read Data	Write Data	Read Status
Quantum[1]	X	X	X	X	X	X	X
984A	X	X	X	X	X	X	X
984B	X	X	X	X	X	X	X
984X	X	X	X	X	X	X	X
584L	X	X	X	X	X	X	X
584M	X	X	X	X	X	X	X

[1]Quantum programmable controllers must be configured as Modicon 984 programmable controllers.

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

2
Installing and Configuring the DAS

This chapter provides the information you need to install the BASEstar Classic 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 installation.

Note

Back up the disks on your system before installing this software. This will provide a method to restore your system in the event of an installation problem. The procedure for backing up disks is described in the *OpenVMS System Management Utilities Reference Manual*.

2.1.1 Hardware

The DAS for Modicon programmable controllers requires the same hardware as BASEstar Classic device connection management software. For specific information on device connect management hardware requirements, refer to the *BASEstar Classic Installation Guide*. Additional hardware needed for the DAS for Modicon programmable controllers includes:

If you intend to use the Modbus communication protocol you must have:

- J478 modem for Modicon 584 programmable controllers
- Cables from the host port or LAT port (see the cable wiring diagram in Appendix A)

If you intend to use the Modbus Plus communication protocol you must have:

- SQ85 Network Card (OpenVMS/VAX systems)
- SA85 Network Card (OpenVMS/Alpha systems)
- Modicon 984 series programmable controllers that end in the number 5 (for example 984/385)

If you intend to use Modbus over TCP/IP communications you must have:

- 140 NOE 211 00 (twisted pair) Modicon Quantum Ethernet TCP/IP Module
- 140 NOE 251 00 (fiber optic) Modicon Quantum Ethernet TCP/IP Module

Quantum programmable controllers are configured as Modicon 984 programmable controllers.

2.1.2 Software

Before installing the DAS for Modicon programmable controllers, the following software must already be installed:

- OpenVMS Version 5.5-2 or higher (VAX)
- OpenVMS Version 6.1 or higher (Alpha)
- BASEstar Classic for OpenVMS, Version 3.4

For information on installing the above software, refer to the *BASEstar Classic 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)

In addition to the above software, if Modbus over 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

Peak/Net Usage	Approximate Space Requirements (Blocks)
Peak usage (during installation)	700 (VAX)
	1100 (Alpha)
Net usage (after installation)	550 (VAX)
	1000 (Alpha)

2.2 Installation Procedure

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

   ```
   $  @SYS$UPDATE:VMSINSTAL DCM_DTLVVA034 ddcu:
   ```

   The **DCM_DTLVVA034** argument is the kit name. The **034** portion of the name is the version number. The **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 7-FEB-2000 at 07:32.

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:

  DCM_DTLVVA V3.4

        Beginning installation of DCM_DTLVVA V3.4 at 07:32

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

   Copyright 1990 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 Classic DAS for Modicon Programmable Controllers
   installation procedure.

   Checking OpenVMS version...

   Now checking that BASEstar Classic is installed...

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

* Does your site have Modbus communications [YES]?
* Does your site have Modbus Plus communications [YES]?
* Does your site have Modbus TCP/IP communications [YES]?
* Does your site have Modicon 584 programmable controllers [YES]?
%DCM_DTLVVA-I-MOD984, Including support for Modicon 984 programmable controllers

   The configuration template file for Modicon support, DCM_MOD_CONFIG.TEMPLATE,
   is used to define the Modicon paths, types, and devices.  Edit this
   file, as necessary, to reflect your specific site configuration.
```

```
   During installation it will be placed in the directory BCC$SYSDATA.

   The installation procedure has no further questions to ask and will
   complete in 1 to 5 minutes depending on the system and system load.

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

   Copyright 1990 Compaq Computer Corporation

   Confidential commercial computer software.  Valid license required.

   Executing the Installation Verification Procedure.

   BASEstar Classic DAS for Modicon Programmable Controllors
   installation procedure has succeeded.

        Installation of DCM_DTLVVA V3.4 completed at 07:33

        VMSINSTAL procedure done at 07:33
```

2.2.1 Files Created During Installation

Table 2–2 lists the files that the DAS for Modicon programmable controllers installation procedure creates and the directories in which they are placed.

Table 2–2 Files Created During Installation

Directory	Filename
BCC$SYSDATA:	DCM_MOD_CONFIG.TEMPLATE
	DCM_MOD$MSG.DAT
SYS$LIBRARY:	ILAN_MOD584.EXE
	ILAN_MOD984.EXE
	ILAN_MODHOST.EXE
	ILAN_MODBUS.EXE
	ILAN_MPLUS.EXE
	ILAN_MODTCP.EXE
SYS$HELP:	DCM_DTLVVA034.RELEASE_NOTES (VAX)
	DCM_DTLVAA034.RELEASE_NOTES (Alpha)
SYS$TEST:	DCM_MOD$IVP.COM

If the BASEstar Classic library system has been installed, the files in Table 2–3 will also be created.

NOTE: The library system is not available on OpenVMS/Alpha systems.

Table 2–3 Files Used by Simple Compare

Directory	Filename
SYS$LIBRARY:	ILAN_MOD584_SCMP.EXE
	ILAN_MOD984_SCMP.EXE

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:

BADBCC, BASEstar Classic software must be installed before DAS for Modicon Programmable Controllers.

Explanation: Error. Incorrect version of or missing BASEstar Classic software.

User Action: Install BASEstar Classic for OpenVMS, Release 3.4 or higher software.

BADDCM, BASEstar Classic Device Connect must be installed before the DAS for Modicon Programmable Controllers.

Explanation: Error. Incorrect version of or missing BASEstar Classic DCM software.

User Action: Install BASEstar Classic DCM for OpenVMS, Release 3.4 or higher software.

BADVMS (VAX), The DAS for Modicon Programmable Controllers must be installed under OpenVMS V5.5-2 or greater.

Explanation: Error. Incorrect version of OpenVMS.

User Action: Install OpenVMS V5.5-2 or higher.

BADVMS (Alpha), The DAS for Modicon Programmable Controllers must be installed under OpenVMS V6.1 or greater.

Explanation: Error. Incorrect version of OpenVMS.

User Action: Install OpenVMS V6.1 or higher.

NETBLOCKS (VAX), The DAS for Modicon Programmable Controllers requires 550 blocks after installation.

Explanation: Error. Not enough disk space to complete installation.

User Action: Delete any unnecessary files, then reinstall.

NETBLOCKS (Alpha), The DAS for Modicon Programmable Controllers requires 1000 blocks after installation.

Explanation: Error. Not enough disk space to complete installation.

User Action: Delete any unnecessary files, then reinstall.

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.

2.3 Postinstallation Tasks

This section describes the tasks to perform after installing the DAS for Modicon programmable controllers, including editing the configuration file, setting the BASEstar Classic DCM support block parameter, configuring ports, addressing a Modbus Plus or TCP/IP node, setting unsolicited parameters and setting up plant-floor equipment.

2.3.1 Editing the Configuration File

A configuration file template, BCC$SYSDATA:DCM_MOD_CONFIG.TEMPLATE, is supplied with the DAS kit. The configuration file contains definitions for types, paths, and devices. A type record represents a protocol emulator (PE). A path record represents a network interface (NI). Copy the template file, edit the file to include site-specific information about types and paths, and execute the file.

The following sections give examples of the type, path, device records. Refer to the *BASEstar Classic Command Line Interface User's Guide* for more information about creating type, path, and device definitions.

2.3.1.1 Editing Type Records

The following example shows the type records created by the configuration file:

```
create type MOD_584_TYPE/manufacturer="Modicon"/model=584 -
  /protocol=MOD584/description="Modicon 584"/LOG

create type MOD_984_TYPE/manufacturer="Modicon"/model=984 -
  /protocol=MOD984/description="Modicon 984"/LOG

create type MOD_HOST_TYPE/manufacturer="Modicon"/model=984 -
  /protocol=MODHOST/description="Modicon 984"/LOG
```

The `/protocol` fields identifies the PE to use in communicating to the device, so this field cannot be changed. The `MODHOST` protocol is used only with Modbus-Plus devices. The model and manufacturer fields are ignored by this DAS, so they can be any value valid to Device Connection Management.

2.3.1.2 Editing Path Records

The following example shows the path records created by the configuration file:

```
create path MODBUS_PATH /vaxport=LTAXXX:/netname=MODBUS -
  /multidrop/io_size=512/retries=2 -
  /line=(mode=rtu,type_ahead,speed=9600,data=8,stop=1)/LOG
modify path MODBUS_PATH/line=(parity=even,full,hangup)/LOG

create path MPLUS_PATH /vaxport=MPA1:/netname=MPLUS -
  /multidrop/LOG

create path MODTCP_PATH /vaxport=UCX$DEVICE:/netname=MODTCP -
  /multidrop/LOG
```

Devices using Modbus communications should use the MODBUS path. Devices using Modbus-Plus communications should use the MPLUS path. Devices using Modbus TCP/IP communications should use the MODTCP path.

If using the Modbus path, the `/vaxport` field refers to the port (either LAT or TT) that is connected to the Modbus line (either directly or through a modem). If a direct connection to the device is made, then the path can be set as `POINT_TO_POINT`. If the connection is to a Modbus network with more than one device configured, then the path must be set as `MULTIDROP`. The `/netname` field identifies the NI that is being used to communicate with the device.

Table 2–4 lists the path parameters for a Modbus connection and indicates the value(s) that are allowed for each.

Table 2–4 Modbus Path Parameters

Parameter	Value(s)
VAXport[1]	LTAXXX:, TTXX:
Netname	MODBUS
Multidrop	MULTIDROP, POINT_TO_POINT
Timeout[2]	1-15
Retries	1-5
IO Size[3]	

[1]This parameter should be set to the OpenVMS device name used to connect to the Modbus network or device.

[2]The timeout defaults to 9 seconds if no timeout is specified.

[3]The IO_SIZE parameter is ignored by this DAS.

Table 2–5 lists the line parameters and indicates the value(s) that are allowed for each. If only one value is allowed, then the DAS will generate an error if the user sets the parameter to anything other than the allowed value. Verify that the VAXport and the line parameters match your plant-floor configuration.

Table 2–5 Modbus Line Parameters

Parameter	Value(s)
Stop Bits	1
Data Bits	7-8
Parity	even, odd, none
Speed	110-19200
Full Duplex	full
Mode	rtu,ascii
Echo	noecho
Host Synch	hostsync, nohostsync
TT Synch	ttsync, nottsync
Typeahead	type_ahead
Hangup	hangup
Modem	modem, nomodem
Dialup	dialup, nodialup
Disconnect	disconnect, nodisconnect

If using the Modbus-Plus path, the `/vaxport` field refers to the OpenVMS device that implements communications to the Modbus-Plus network (MPA1-MPA4). Up to four Modbus-Plus communications cards are allowed on one system. On Alpha systems, the name of the OpenVMS dvice can be JPA1-JPA4 due to a conflict in device names on some OpenVMS systems. The path must be set as `MULTIDROP` since Modbus-Plus is a multidrop network. The `/netname` field identifies the NI that is being used to communicate with the device.

Table 2–6 lists the path parameters for a Modbus-Plus connection and indicates the value(s) that are allowed for each.

Table 2–6 Modbus-Plus Path Parameters

Parameter	Value(s)
VAXport[1]	MPA1-MPA4, JPA1-JPA4
Netname	MPLUS
Multidrop	MULTIDROP
Timeout[2]	1-15
Retries[3]	
IO Size[4]	

[1]This parameter should be set to the OpenVMS device name used to connect to the Modbus-Plus network. JPA1-JPA4 are used only on some OpenVMS Alpha versions where there is a conflict with the MPA1-MPA4 device names.

[2]The timeout defaults to 9 seconds if no timeout is specified.

[3]The parameter is ignored by this DAS. The DAS does not do retries.

[4]The IO_SIZE parameter is ignored by this DAS.

If using the Modbus TCP/IP path, the `/vaxport` field refers to the logical name that points to the OpenVMS internet pseudodevice that implements communications to the TCP/IP network. UCX$DEVICE is used on systems that have UCX installed. TCPIP$DEVICE may be used on systems that have the newer TCPIP interface installed. The path must be set as `MULTIDROP`, since there is only one pseudodevice defined, even though all connections are point-to-point connections. The `/netname` field identifies the NI that is being used to communicate with the device.

Table 2–7 lists the path parameters for a Modbus TCP/IP connection and indicates the value(s) that are allowed for each.

Table 2–7 Modbus TCP/IP Path Parameters

Parameter	Value(s)
VAXport[1]	UCX$DEVICE TCPIP$DEVICE
Netname	MODTCP
Multidrop	MULTIDROP
Timeout[2]	1-15
Retries[3]	
IO Size[4]	

[1]UCX$DEVICE is used on systems that have UCX installed. TCPIP$DEVICE may be used on systems that have the newer TCPIP interface installed.

[2]The timeout defaults to 9 seconds if no timeout is specified.

[3]The parameter is ignored by this DAS. The DAS does not do retries.

[4]The IO_SIZE parameter is ignored by this DAS.

2.3.1.3 Editing Device Records

The following example shows the device records created by the configuration file:

```
create device MOD_584 /type=MOD_584_TYPE -
  /descrip="Modicon 584"/netaddress=YY -
  /path=MODBUS_PATH/manuf="Modicon"/timeout=300/LOG

create device MOD_984 /type=MOD_984_TYPE -
  /descrip="Modicon 984"/netaddress=YY -
  /path=MODBUS_PATH/manuf="Modicon"/timeout=400/LOG

create device MPLUS_984 /type=MOD_984_TYPE -
  /descrip="Modicon 984"/netaddress="YY-YY-YY-YY-YY" -
  /path=MPLUS_PATH/manuf="Modicon"/timeout=400/unsolicited/LOG

create device MHOST_984 /type=MOD_HOST_TYPE -
  /descrip="Modicon 984"/netaddress="YY" -
  /path=MPLUS_PATH/manuf="Modicon"/timeout=400/unsolicited/LOG

create device MODTCP_984 /type=MOD_984_TYPE -
  /descrip="Quantum TCP/IP"/netaddress="YY.YY.YY.YY" -
  /path=MODTCP_PATH/manuf="Modicon"/timeout=400/unsolicited/LOG
```

The timeout on the device definition controls the time that BASEstar Classic device connection management allows for a device operation to complete. The value for the device timeout should be larger than the expected time of the longest device operation and also larger than the timeout on the path definition. To control the timeout for individual I/O operations to the device, use the timeout on the path definition.

Unsolicited data collection is supported only for the Modbus-Plus and TCP/IP NIs, so the `/unsolicited` qualifier should be used only on devices with a Modbus-Plus or TCP/IP path.

The `MODHOST` PE is supported for use only with the Modbus-Plus NI and is used for unsolicited data collection. See Section 3.4.2 for more information.

2.3.1.4 Network Address Using Modbus

Modbus addresses are decimal numbers from 1-255.

2.3.1.5 Network Address Using Modbus Plus

Modbus Plus requires 5 bytes of routing information to address a particular node on the network. This method allows remote nodes to be addressed up to four networks away from the host. The range of node addresses on a single network is 1 ... 64. The format for the network address field within DCM is as follows:

```
xx-xx-xx-xx-xx
```

Where **xx** is the routing address byte. All five bytes of routing information must be included in the network address field. A zero as a routing byte indicates that no further routing will occur. Once a zero is encountered as a routing address byte all of the remaining routing bytes must be zero.

2.3.1.6 Network Address for the Modbus Plus Host

When addressing the host node only one routing address byte is required. Specify the Modbus-Plus address of the host SQ-85/SA-85 card. Only one host device is allowed per Modbus-Plus card.

2.3.1.7 Network Address Using TCP/IP Ethernet Interface Module

To address a TCP/IP NOE module use the TCP/IP address of the ethernet interface of the module. The format for the network address field within DCM is as follows:

```
xx.xx.xx.xx
```

Where **xx** is the routing sub-net byte. All four bytes of routing information must be included in the network address field. The DAS only supports sub-net routing format. Using a host name rather than a sub-net address is not supported.

For more information about maintaining type, path, and device definitions, refer to the *BASEstar Classic Command Line Interface User's Guide*.

2.3.2 Setting the SPT Block Parameter

The ILAN$MAX_SPT_REQUESTS parameter specifies the total number of blocks that can be allocated in the SPT (support) global section. DASes use blocks in the global section for storing data structures and for doing device I/O. The SPT global section is sized by calculating the number of SMALL, MEDIUM, LARGE and EXTRA LARGE blocks that the section should contain. Some blocks remain for the life of a device and some are allocated and deallocated for each I/O operation. Table 2–8 shows the static blocks of each size that are used by the DAS.

Table 2–8 SPT Static Block Sizes

PE/NI	Block Size	Quantity[1]	Block Type
PE	MEDIUM	1	Device
		7[2]	Device[3]
	SMALL	1	Device[3]
Modbus NI	MEDIUM[4]	1	Line
	SMALL	1	Device
		1	DAS
Modbus-Plus NI	LARGE	2[5]	Device
	MEDIUM	1	Line
	SMALL	1	Device
		7[2]	Device[3]
		1	DAS
TCP/IP NI	LARGE	1	Device
	MEDIUM	1	DAS
		7[2]	Device

[1]Quantity is quantity per device, per line, etc.

[2]Number of blocks is = ILAN$MODXXX_UNSOL_QUEUE_SIZE.

[3]Only created if the device is marked "unsolicited".

[4]This block is LARGE if ASCII mode is used.

[5]Number of blocks is number of slave paths allocated.

Table 2–9 shows the dynamic blocks of each size that are used by the DAS. These blocks are created and deleted as the device does I/O. I/O can be initiated solicitedly by a user or a polling set or can be initiated unsolicitedly by the device.

Table 2–9 SPT Dynamic Block Sizes

PE/NI	Block Size	Quantity[1]	I/O Type
PE	MEDIUM	1	Any
Modbus NI	MEDIUM[2]	1	Any
Modbus-Plus NI	LARGE	1	Any
TCP/IP NI	MEDIUM	1	Any

[1]Quantity is quantity per I/O.

[2]This block is LARGE if ASCII mode is used.

The size of the SPT global section can be tuned by changing the percentage of each kind of block that is created. Refer to the *BASEstar Classic Configuration and Tuning Guide* for instructions on changing the percentage of each size of block that is created in the global section.

2.3.3 Configuring Ports

A device can be physically connected to a local area transport (LAT) port. To use a LAT port with the DAS, you must define the LAT port on the host system by using the LAT control program. The following DCL example shows how to define **port_2** on the LAT node **srvr4** as **LTA992**. It is very important when defining the LAT port to use the **/NOQUEUE** option. If the port is owned by another process device connection management reports an error during the establishment of a connection to the device. When you define a path and a device to BASEstar Classic device connection management LTA992 is supplied as the VAXport.

```
$ MCR LATCP

LCP> CREATE PORT LTA992

LCP> SET PORT LTA992 /NODE=srvr4/PORT=port_2/NOQUEUE
```

For the DAS to send and receive data correctly, you must set and define LAT port characteristics to match the line parameters of the path definition for devices and terminals connected to the LAT. An example of the characteristics set at the LAT prompt follows:

```
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
```

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	1024

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

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

2.3.4 Setting DAS-Specific Parameters

The following sections describe how to set the DAS-specific BASEstar Classic parameters.

None of the parameters created for the DAS for Modicon programmable controllers are dynamic. BASEstar Classic device connection management must be shut down and started again before parameter modifications become effective.

2.3.4.1 Setting Modbus Parameters

The installation procedure for the DAS for Modicon programmable controllers creates BASEstar Classic parameter(s) to allow you to tune the DAS environment for your Modbus network. These parameter(s) are listed in Table 2–10.

Table 2–10 Modbus Parameters

Parameter	Default	Description
ILAN$MODBUS_TRANSMIT_DELAY	0 00:00:00.00[1]	This value specifies the time to delay after receiving a response to a Modbus message before sending the next message to a new address.

[1]The value of this parameter is dependent on the baud rate and should be set to at least three and one-half character times. The lowest value that this parameter can be set to is 00:00:00.01 which is an acceptable value for baud rates of 4800 baud and higher.

Modbus requires an inter-message delay and on fast systems, normal processing overhead does not provide enough delay resulting in intermittent timeouts. This problem is most likely to occur on systems with directly connected ports (not LAT). The ILAN$MODBUS_TRANSMIT_DELAY parameter provides this delay if the parameter is not set to 0 (the default).

2.3.4.2 Setting Unsolicited Parameters

The installation procedure for the DAS for Modicon programmable controllers creates BASEstar Classic parameters to allow you to tune the DAS environment for collecting unsolicited data. These parameters are listed in Table 2–11.

Table 2–11 Unsolicited Parameters

Parameter	Default	Description
ILAN$MOD984_MAX_REG_ADDRESS[1]	1000	This value specifies the maximum holding register offset value that can be specified as the destination in a Modbus MSTR write function block.
ILAN$MOD984_UNSOL_QUEUE_SIZE[1]	7	This value specifies the size of the queue to receive unsolicited write messages from the Modbus-Plus or TCP/IP network. A larger value allows more data to be handled in a burst of messages from the Modbus-Plus or TCP/IP network at the expense of more memory consumption.
ILAN$MODHOST_MAX_REG_ADDRESS[2]	1000	This value specifies the maximum holding register offset value that can be specified as the destination in a Modbus MSTR read function block.
ILAN$MODHOST_UNSOL_QUEUE_SIZE[2]	7	This value specifies the size of the queue to receive unsolicited read messages from the Modbus-Plus network. A larger value allows more data to be handled in a burst of messages from the Modbus-Plus network at the expense of more memory consumption.
ILAN$MODHOST_UNSOL_WRITE[2]	0	This value specifies whether unsolicited writes are sent to the MODHOST device or to the device identified by the first five words of the message. If this parameter is "0", then routing information is expected to be in the first five words of the message and the write is sent to the device with a matching network address.

[1]Modbus Plus or TCP/IP networks.
[2]Modbus Plus networks only.

2.3.4.3 Setting TCP/IP Parameters

The installation procedure for the DAS for Modicon programmable controllers creates BASEstar Classic parameters to allow you to tune the TCP/IP environment. These parameters are listed in Table 2–12.

Table 2–12 TCP/IP Parameters

Parameter	Default	Description
ILAN$MODTCP_PROBE_TIME[1]	10	This value specifies the time in seconds that TCP/IP will use to probe a for a broken connection.

[1]The probe time plus drop time is the total time that will elapse between when a connection is physically broken until the DAS will detect that the connection no longer exists.

(continued on next page)

Table 2–12 (Cont.) TCP/IP Parameters

Parameter	Default	Description
ILAN$MODTCP_DROP_TIME[2]	30	This value specifies the time in seconds after a broken connection has been detected until the connection is actually dropped.

[2]The drop time must be greater than the probe time or a bad parameter error will be returned when trying to set the socket option. The drop time should be an even multiple of the probe time.

For information on configuring and setting up Digital TCP/IP Services for OpenVMS see *Digital TCP/IP Services for OpenVMS Installation and Configuration*.

2.3.5 Setting Up Plant-Floor Equipment

For information on configuring your Modicon equipment, see Appendix A. For information on your specific device, refer to the *Modbus System Planning Guide* and the *J478 Stand-Alone Modem Guide* for Modbus communications, to the *Modicon DEC Host Based Devices User's Guide* for Modbus-Plus communications or to the *Modicon Quantum Ethernet TCP/IP Module User Guide* for Modbus TCP/IP communications.

2.4 Tracing Device Communications

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

To enable tracing do the following:

- Set one of the following logicals to 1:

DCM_MOD$MODBUS_TRACE	Trace the MODBUS NI (Network Interface)
DCM_MOD$MPLUS_TRACE	Trace the MPLUS NI (Network Interface)
DCM_MOD$MODTCP_TRACE	Trace the MODTCP NI (Network Interface)

  The logical name must be defined in a scope that will be seen by the device server process (ILAN$DEVSRV).

- Set the trace output logical to a file or other output device:

DCM_MOD$MODBUS_TRACE_OUTPUT	Trace the MODBUS NI (Network Interface)
DCM_MOD$MPLUS_TRACE_OUTPUT	Trace the MPLUS NI (Network Interface)
DCM_MOD$MODTCP_TRACE_OUTPUT	Trace MODTCP NI (Network Interface)

  The logical name must be defined in a scope that will be seen by the device server process (ILAN$DEVSRV).

- Start the tracing.

  To start tracing for the NI, disable all devices on the line for which tracing is desired and reenable them. The trace logical is read when the first device on the line is enabled.

The trace output is flushed to the file after every 100 lines that are traced, so trace output may lag actual device I/O.

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

2.5 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 Classic 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 DAS Functions

DAS for Modicon programmable controllers functions are accessed through the BASEstar Classic software, through the following means:

- Commands
- Menu system
- Callable services

To use the BASEstar Classic device connection management commands, enter the following command at the DCL prompt ($):

```
$  BSTAR DCM
```

To use the BASEstar Classic device connection management menu system, enter the following command:

```
$  BSTAR/MENU
```

For information about the BASEstar Classic device connection management commands and the menu system, refer to the *BASEstar Classic Menu Interface User's Guide*.

For information about the BASEstar Classic device connection management callable services, refer to the *BASEstar Classic Introduction to Callable Services*.

3.2 Structured Data

Structured data is a block of data, typically represented by a buffer, that can be broken down into meaningful elements of data. Structured data allows you to impose a structure on the data formats for any given read or write operation. A structure tells the device how to interpret the data that it is reading or writing.

Structured data can represent synchronized data elements from a single source, such as a plant-floor device. Structured data can also represent composite views of different data elements from different sources, such as the total part counts from several different plant-floor devices.

For the DAS for Modicon programmable controllers, mixed data types are not allowed within a given address range.

3.3 Supported Functions

This section describes the functions that are supported by the DAS for Modicon programmable controllers devices.

Table 3–1 displays the Modicon devices and the functions they support.

Table 3–1 Modicon Devices and Functions

Device	Upload	Download	Start	Stop	Read Data	Write Data	Read Status
Quantum[1]	X	X	X	X	X	X	X
984A	X	X	X	X	X	X	X
984B	X	X	X	X	X	X	X
984X	X	X	X	X	X	X	X
584L	X	X	X	X	X	X	X
584M	X	X	X	X	X	X	X

[1]Quantum programmable controllers must be configured as Modicon 984 programmable controllers.

3.3.1 Start and Stop

The start and stop functions change the operating mode of a device. The start function enables outputs and the stop function disables outputs.

Warning

Devices can control complex and perhaps dangerous industrial processes. Do not use the start and stop functions until you take the required safety precautions and put required operating restrictions into effect. Refer to the manufacturer's documentation for specific safety precautions.

3.3.2 Upload and Download

The upload function transfers the contents of a device's memory to an OpenVMS file. The download function transfers the contents of an OpenVMS file to a device's memory.

To perform the download function, the memory protect keyswitch must be in the OFF position for all device models. Also, the DAS must log in to the programmable controller. If another user is already logged in the download will fail.

Note

You perform a download function on a Modicon device without first stopping the device.

3.3.3 Read and Write Data Functions

Use the BASEstar Classic device connection management read data and write data functions to read data from and write data to a specific address in device storage. Address syntax is specific to each device or device family, and is similar to the syntax used with the Modicon terminals. The supported data types depend on the device.

Example 3–1 shows an example of the information displayed when you enter the READ DATA command.

Example 3–1 Read Data Screen

```
DCM> read data mod_584/addr=00001/form=bit

Device      : MOD_584
Data starting at address 00001 for BIT  element.
Data:
    0:      1

DCM> read data mod_584/addr=00001/form=array[64]:bit

Device      : MOD_584
Data starting at address 00001 for ARRAY[64]:BIT  element.
Data:
    0:   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   16:   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   32:   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   48:   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Example 3–2 provides an example of the information displayed when you enter the READ DATA command using structured data.

Example 3–2 Read Data Screen with Structured Data

```
DCM> cre phyp p1/dev=mod_584_1/addr=00001/form="struct" -
DCM> (word,word,array[10]:word)"/write
DCM> read phyp p1/dev=mod_584_1

Point       : P1
Device      : MOD_584_1
Address     : 40100
Format      : STRUCTURE(S_WORD,S_WORD,ARRAY[10]:S_WORD)

Data:
    0       :       26
    1.0     :       29      29      29      29      29      29
    1.7     :       29      29      29
```

Example 3–3 shows an example of the information displayed when you enter the WRITE DATA command.

Example 3–3 Write Data Screen

```
DCM> write data mod_584/addr=40200/format=word

Device      : MOD_584
Data starting at address 40200 for S_WORD element

Data value           0 : 100
```

Example 3–4 shows an example of the information displayed when you enter the WRITE DATA command using structured data.

Example 3–4 Write Data Screen with Structured Data

```
DCM> write phyp p2/dev=mod_584_1

Point       : P2
Device      : MOD_584_1
Address     : 00001
Format      : STRUCTURE(BIT,BIT,ARRAY[8]:BIT)

Data value     0       :        1
Data value     1       :        0
Data value     2.0     :        1
Data value     2.1     :        1
Data value     2.2     :        1
Data value     2.3     :        0
Data value     2.4     :        0
Data value     2.5     :        0
Data value     2.6     :        0
Data value     2.7     :        1
```

The following sections describe the use of the read data and write data functions by specific Modicon devices, including addressing and data type formats. The supported data type formats are bit, word, unsigned word, and longword.

3.3.3.1 Quantum, Modicon 984 and Modicon 584

Addressing and data type formats are necessary for performing the read data and write data functions for the Modicon 584, 984 and Quantum devices.

Addressing

The valid addressing types for the Modicon 584, 984 and Quantum are:

```
00001 - 0xxxx          Coils
10001 - 1xxxx          Discrete inputs
30001 - 3xxxx          Input registers
40001 - 4xxxx          Holding registers
```

Note

For the Modicon 984 and Quantum, the maximum holding and input register is 65535. For the Modicon 584, the maximum holding and input register is 9999.

For controllers that have the extended file capability, the format is as follows:

```
60000:1 - 6xxxx:y      xxxx is the highest register number within a file
                       y    is the file number
```

Note

You cannot write to discrete inputs or input registers.

Data Type Formats

The valid data type formats for the Modicon 584, 984 and Quantum are:

```
00001 - 0xxxxx          Bit, Word or U_word
10001 - 1xxxxx          Bit, Word or U_word
30001 - 3xxxxx          Bit, Word or U_word
40001 - 4xxxxx          Bit, Word, U_word, Long, U_long or Float
60001:1 - 6xxxxx:y      Bit, Word or U_word
```

Note

When writing bits to word based registers, a multiple of 16 bits must be written.

3.3.4 Read Status

The read status function issues a diagnostic status request to the programmable controller, interprets the device response, and returns the interpretation as a character buffer.

The *BASEstar Classic Command Line Interface User's Guide* gives detailed information about the READ STATUS command. To display the returned values for device status, enter the following at the BASEstar Classic device connection management prompt:

```
DCM> READ STATUS device-name/FULL
```

Example 3–5 shows an example of the values returned in the status buffer when you enter the command for reading device status.

Example 3–5 Read Status Screen

```
DCM> read status mod_984/full

 Status of device mod_984 at dd-mmm-yyyy 09:50:54.72

  Processor Type:       Modicon 984
  Processor State:      STOPPED
  Memory Protection:    OFF
  Battery Status:       LOW

  Machine Stop Status:
    o Peripheral Port STOP State
    o No one is currently logged in.
```

3.3.4.1 Unsolicited Counters

For the host pseudo-device used to process unsolicited read and write requests from a Modbus-Plus device the read status command displays the number of read requests, the number of read request errors, the number of "trigger" notification sent, the number of write requests and the number of write request errors. "Triggers" and unsolicited read and write requests are discussed in Section 3.4.2.

For Modicon 984 and Quantum programmable controllers that support unsolicited data collection, the read status command also displays the information displayed for the host pseudo-device as well as the information displayed in Example 3–5. Unsolicited read and write requests are discussed in Section 3.4.2.

To reset the unsolicited counters maintained for the Modicon 984 or Quantum, use the "RESET" qualifer when reading status from the device.

```
DCM> read status mod_984/full/qualifier=reset

 Status of device  mod_984  at dd-mmm-yyyy 13:21:34.91

  Unsolicited read count         :0
  Unsolicited read err count     :0
  Unsolicited read trigger count :0
  Unsolicited write count        :0
  Unsolicited write err count    :0

  Processor Type:       Modicon 984
  Processor State:      RUNNING
  Memory Protection:    OFF
  Battery Status:       GOOD

    o No one is currently logged in.
```

3.4 Automatic Data Collection

BASEstar Classic device connection management supports automatic data collection using both polled and unsolicited data collection. Polled data collection is supported for Modbus, Modbus-Plus and TCP/IP communications. Unsolicited data collection is supported for Modbus-Plus and TCP/IP data collection only.

3.4.1 Polled Data Collection

BASEstar Classic device connection management physical points may be grouped together into pollsets to optimize data collection. Pollsets created for the Modicon devices have the following limitations:

- Physical points of differing data types cannot be part of the same pollset. For example, a physical point of type WORD and a physical point of type BIT cannot be members of the same pollset. Physical points of the same data type with varying element counts can reside in the same pollset.

For more information on physical points, refer to the *BASEstar Classic Command Line Interface User's Guide*.

3.4.2 Unsolicited Data Collection

The DAS for Modicon programmable controllers supports unsolicited data collection using Modbus-Plus or TCP/IP communications only. Unsolicited data collection for the Modicon devices have the following limitations:

- The only supported functions are reads and writes to the host. Get statistics, etc. are not supported.

- Only holding registers can be read from or written to the host. This limitation is imposed by the MSTR function block programming.
- If the size of the data being written is greater than the size of the corresponding physical point, the truncated data is still written, but the point is given a truncation warning status.
- If the size of the data being read is greater than the size of the corresponding physical point, then the amount of data available is returned to the device and a message is logged to the history file.

When using Modbus-Plus communications, the following limitations also apply:

- Reads or writes must be done to slave path numbers 1 or 2. Other slave paths are not supported.
- For writes, the first 5 words written must contain routing information for the node performing the write. The DAS for Modicon programmable controllers uses the first five words of the message to determine which programmable controller sent the message and therefore which device to associate with this message.

  Writes can be also be sent to a special "host" device if the parameter ILAN$MODHOST_UNSOL_WRITE is set to a "1". This parameter affects all devices on a system and is read by the DAS when the device is enabled.
- For reads, a special "host" device must be used to receive the unsolicited read request.

3.4.2.1 Unsolicited ID

The DAS for Modicon programmable controllers supports both unsolicited read and write requests from the programmable controller. The following sections describe the unsolicited ID syntax for reads and writes.

Unsolicited IDs for Write

For writes, the unsolicited ID specified in the physical point definition defines the holding register being written by the MSTR function block.

For Modbus-Plus the first five words of the message must contain the routing information for the programmable controller sending the data. The actual data starts after the routing information. Note that the routing information is not passed on to BASEstar DCM and is used only by the DAS. If the parameter ILAN$MODHOST_UNSOL_WRITE is set to a "1", then the write is sent to the "host" device and the DAS interprets the entire message as data.

For TCP/IP no routing information is included in the message so the actual data starts at the holding register specified in the unsolicited ID.

The valid unsolicited ID type for writes is:

```
400001 - 4xxxxx          Word or U_word
```

The maximum holding register is specified by the parameter ILAN$MOD984_ MAX_REG_ADDRESS.

Unsolicited IDs for Read

For reads, the unsolicited ID specified in the physical point definition defines the holding register being read by the MSTR function block. It also defines the name of the logical point from which to get the data to be returned to the device.

The valid unsolicited ID types for reads are:

```
400001:POINT_NAME - 4xxxxx:POINT_NAME          Word or U_word
400001:TRIGGER - 4xxxxx:TRIGGER                Bit
```

The POINT_NAME is the name of a BASEstar Classic logical point from which to retrieve the data to be sent to the device.

For Modbus-Plus the maximum holding register is specified by the parameter ILAN$MODHOST_MAX_REG_ADDRESS.

For TCP/IP the maximum holding register is the same as that used for writes (ILAN$MOD984_MAX_REG_ADDRESS).

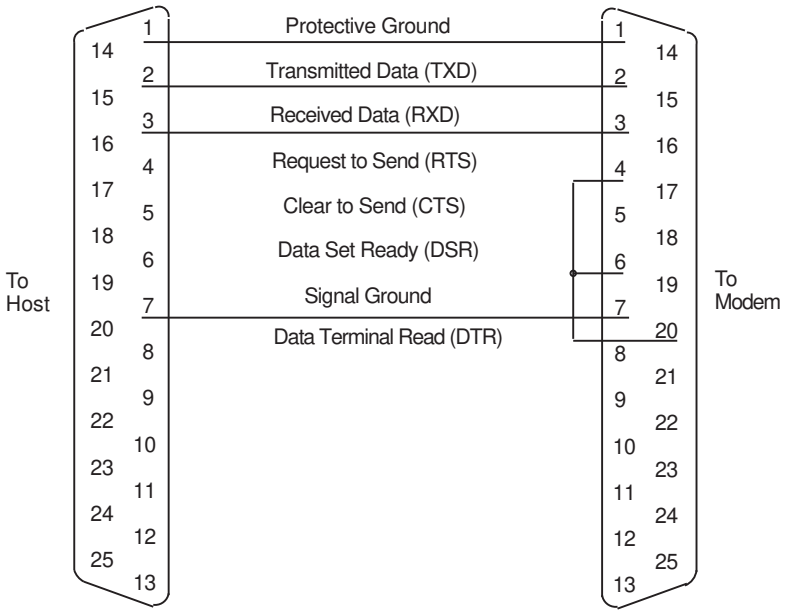
A trigger unsolicited ID can also be specified. The trigger point can be used to notify an application when the programmable controller has read the associated unsolicited read point. A trigger is specified by creating a physical point with an unsolicited ID with the same holding register number of the associated physical point and a "POINT_NAME" of "TRIGGER". The "TRIGGER" keyword is reserved and cannot be used as a logical point name.

A

Equipment Configuration

Figure A–1 shows the cable wiring and pin assignments from the processor to the J478 modem.

Figure A–1 Processor-to-Modem Connection



Equipment Configuration

Modbus Configuration

Modbus is an industrial communication network that can link one master to as many as 247 slaves. Specific masters may have restrictions that limit the number of slaves.

The host port modem is wired as the master on the Modbus. The master modem is connected by twisted-pair wire to all of the slave modems, such as a J478, that interface to the individual programmable controllers.

Figure A–2 represents a Modbus configuration.

Figure A–2 Modbus Block Diagram

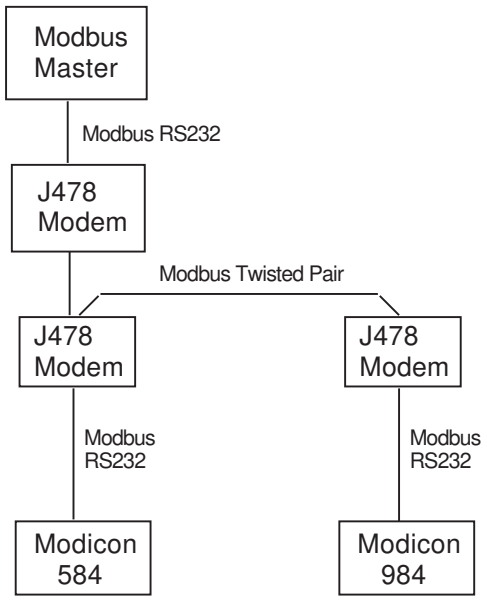
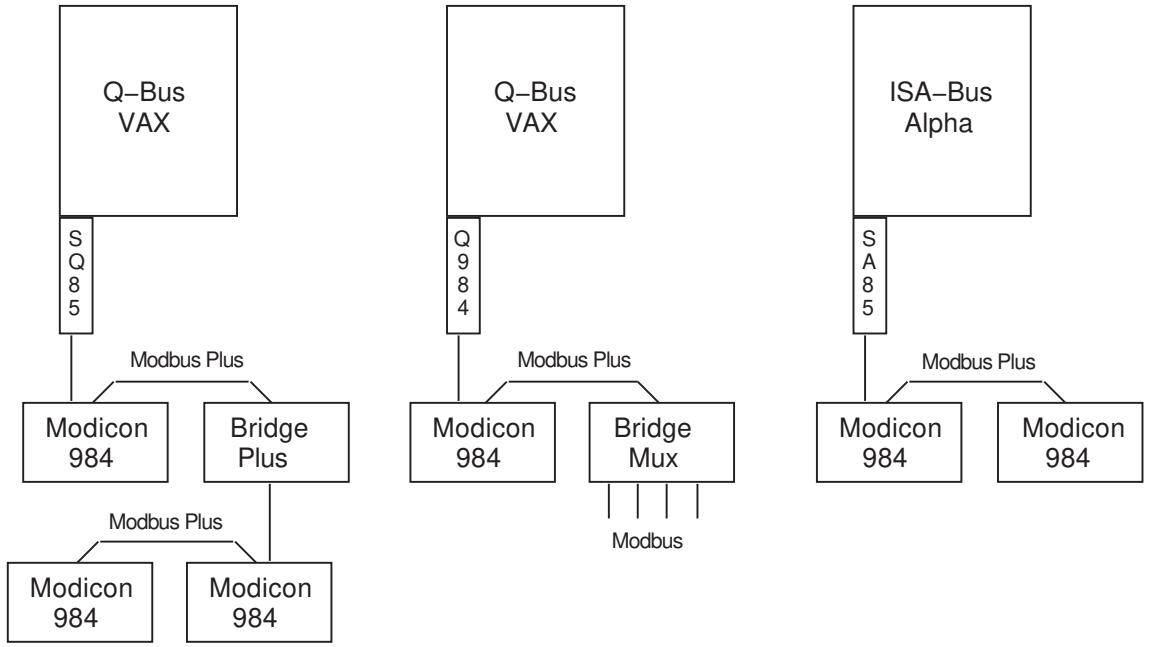


Figure A–3 represents a Modbus-Plus configuration.

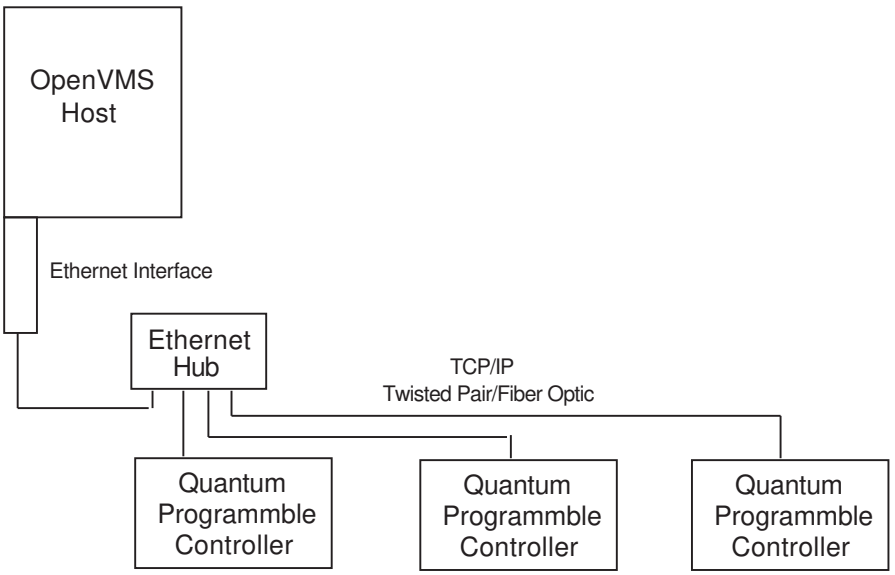
Figure A–3 Modbus Plus Block Diagram



Equipment Configuration

Figure A–4 represents a Modbus TCP/IP configuration.

Figure A–4 Modbus TCP/IP Block Diagram



B

Logged Messages

The messages in the following sections are logged to the BASEstar Classic history file by the DAS for Modicon programmable controllers. These messages are logged to provide more detailed diagnostic information than what is supplied by the returned status values.

Messages logged to the history file for DAS for Modicon programmable controllers use event class 21 and event type 24.

To view all messages logged by this DAS use the following syntax:

```
$ BSTAR
BSTAR> SHOW HISTORY/EVENT=21.24.*
```

B.1 Modbus NI Logged Messages

The following messages are logged by the Modbus Network Interface (NI).

Disconnected station #!SL on port !AD.

Explanation: Informational. The device at the port and station address has successfully been disabled.

User Action: None.

Hang up message received for port !AD.

Explanation: Informational. If the port is a LAT port, then either the network connection to the terminal server has been broken or the serial line connection to the PLC has been disconnected. If the port is a TT port, then the serial connection to the PLC has been disconnected.

User Action: If the cause of the hangup was a network disconnection, then determine the cause of the network disconnection (could be physical problems with the network or the terminal server was powered down) and correct the problem. If the cause of the hangup was the port, then check for broken or missing cables and correct any problems found.

Station #!SL is being removed.

Explanation: Informational. The station is being disconnected due to a hangup on the line.

User Action: None.

QIO error.

Explanation: Error. An error occurred when reading or writing data to the port.

User Action: Examine the error message and take action based on the error found. If connected via a terminal server examine the terminal server counters for errors as well. Check cabling for improper length, improper grounding, breaks, etc.

Purged !UW bytes from port !AD.

Explanation: Informational. The current response from the port is being discarded because a different device responded to the request than the one requested, or the checksums do not match.

User Action: If connected via a terminal server examine the terminal server counters for errors. Check cabling for improper length, improper grounding, breaks, etc.

Unexpected reply from station #!UB, Expected station #!UB.

Explanation: Error. A different programmable controller responded to a request than the one expected. This problem could be the result of a faulty controller or because of a faulty cable.

User Action: Check cabling for improper length, improper grounding, breaks, etc. If cabling checks out, then replace the controller.

Checksums did not match - retries exceeded.

Explanation: Error. The received checksum did match the calculated value and retries have been exceeded.

User Action: Check cabling for improper length, improper grounding, breaks, etc. If using a terminal server, check the port counters for errors.

B.2 Modbus Plus NI Logged Messages

The following messages are logged by the Modbus Plus Network Interface (NI).

Disconnected station !AD.

Explanation: Informational. The DAS has successfully aborted the connection to the device.

User Action: None.

Routing byte !UL value !AD invalid.

Explanation: Error. The indicated part of the route is not a number or is not a valid number.

User Action: Modify the network address to be a valid sequence of numbers in xx-xx-xx-xx-xx format.

QIO error when reading slave command for path !UB.

Explanation: Error. A QIO error occurred when reading from the indicated slave path. A slave path is used for reading unsolicited commands (read or write) from the device.

User Action: Examine the error returned and take action based on the error found.

QIO error when writing slave response for path !UB.

Explanation: Error. A QIO error occurred when writing to the indicated slave path. The error occurred when attempting to respond to a command that had been sent to the host.

User Action: Examine the error returned and take action based on the error found.

QIO error when writing master command for path !UB.

Explanation: Error. A QIO error occurred when writing to the indicated master path. A master path is used for writing solicited commands (read or write) to the device.

User Action: Examine the error returned and take action based on the error found.

QIO error reading master response for path !UB.
Node type: !UB, error condition: !UB.

Explanation: Error. A QIO error occurred when reading from the indicated master path. A master path is used for writing solicited commands (read or write) to the device. See *Modicon DEC Host Based Devices User's Guide* for a description of error condition codes.

User Action: Examine the error returned and take action based on the error found.

Insufficient routing bytes in slave write request from source address !UB.

Explanation: Error. The DAS has been configured to expect routing bytes in the message sent on a slave path to the host and less than 5 words of data were sent. The source address is valid only if the programmable controller is on the closest Modbus-Plus network.

User Action: Correct the program in the programmable controller to send the required routing information as well as the data or reconfigure the DAS to not expect routing information in the data. See Section 2.3.4.2 for information on configuring the DAS to not expect routing information in the data.

Rejected slave write request from undefined slave network address !AD.

Explanation: Error. A slave write request was received that was assumed to contain routing information in the first five words, but the routing information did not match any configured device.

User Action: Modify the programmable controller program to identify its own address in the first five words of data that are sent or reconfigure the DAS to not expect routing information in the data. See Section 2.3.4.2 for information on configuring the DAS to not expect routing information in the data.

No unsolicited response block available, request discarded.

Explanation: Error. A request from the device was received, but the queue of unsolicited I/O blocks has been exhausted.

User Action: Increase the size of the queue by increasing the value of the parameter ILAN$MOD984_UNSOL_QUEUE_SIZE or ILAN$MODHOST_UNSOL_QUEUE_SIZE, as appropriate. See Section 2.3.4.2 for more information on these parameters.

Rejected slave read request from source address !UB, no host device defined.

Explanation: Error. A read request was received, but no host device has been configured. It is also possible that the request was received soon after startup and the DAS has not yet been informed of the host device. The source address is only valid for the closest Modbus Plus network.

User Action: Configure a host device to receive unsolicited read requests.

Rejected slave write request from source address !UB, no host device defined.

Explanation: Error. A write request was received, but no host device has been configured. It is also possible that the request was received soon after startup and the DAS has not yet been informed of the host device. The source address is only valid for the closest Modbus Plus network.

User Action: Configure a host device to receive unsolicited write requests or reconfigure the DAS to not send write requests to the host device. See Section 2.3.4.2 for information on configuring the DAS to not send data to the host device.

Rejected function code !UB from source address !UB, unsupported function.

Explanation: Error. A modbus function other than "READ HOLDING REGISTERS", function code "3", or "PRESET MULTIPLE REGISTERS", function code "16", was received by the DAS. These two modbus functions are the only two functions supported by the DAS. The source address is only valid for the closest Modbus Plus network.

User Action: Reprogram the programmable controller to send a supported modbus function code.

HOST address !AD already configured (only one host allowed).

Explanation: Error. A host device has already been configured for this Modbus-Plus communications card. Only one host device is allowed per Modbus-Plus communications card.

User Action: Delete this device.

B.3 Modbus TCP/IP NI Logged Messages

The following messages are logged by the Modbus TCP/IP Network Interface (NI).

Sub-net byte #!UB is invalid.

Explanation: Error. The indicated part of the sub-net address is not a number or is not a valid number.

User Action: Modify the network address to be a valid sequence of numbers in xx.xx.xx.xx format.

A device with this network address has already been defined.

Explanation: Error. An attempt was made to define duplicate network addresses.

User Action: Either delete the other device with the same address or use a different network address for this device.

Error when disconnecting from a socket.

Explanation: Error. An error occurred when attempting to disconnect from the socket using the QIO function IO$_DEACCESS | IO$M_SHUTDOWN.

User Action: Examine the error to determine the cause of the failure and take action based on the error found. See *Digital TCP/IP Services for OpenVMS System Services and C Socket Programming* for information on Digital TCP/IP Services for OpenVMS services.

Error when creating a socket.

Explanation: Error. An error occurred when creating a socket using the QIO function IO$_SETMODE.

User Action: Examine the error to determine the cause of the failure and take action based on the error found. See *Digital TCP/IP Services for OpenVMS System Services and C Socket Programming* for information on Digital TCP/IP Services for OpenVMS services.

Error when setting a socket option.

Explanation: Error. An error occurred when setting a socket option using the QIO function IO$_SETMODE. Socket options being set are PROBE_TIME and DROP_TIME.

User Action: Verify that the values in the BASEstar Classic parameters ILAN$MODTCP_PROBE_TIME and ILAN$MODTCP_DROP_TIME are valid values. See Section 2.3.4.3 for a discussion of these parameters. See *Digital TCP/IP Services for OpenVMS System Services and C Socket Programming* for information on Digital TCP/IP Services for OpenVMS services.

Error when connecting to a socket.

Explanation: Error. An error occurred when connecting to a socket using the QIO function IO$_ACCESS.

User Action: Examine the error to determine the cause of the failure and take action based on the error found. See *Digital TCP/IP Services for OpenVMS System Services and C Socket Programming* for information on Digital TCP/IP Services for OpenVMS services.

Rejecting connection from !UB.!UB.!UB.!UB.

Explanation: Error. The DAS has just received a connection request and there is no device defined with the network address of the incoming connection.

User Action: Create and enable a device with the network address of the incoming connection or reprogram the programmable controller with that address so that it no longer tries to connect to the host.

Write QIO error when writing to a socket.

Explanation: Error. An error occurred when writing to a TCP/IP socket.

User Action: Examine the error to determine the cause of the failure and take action based on the error found. See *Digital TCP/IP Services for OpenVMS System Services and C Socket Programming* for information on Digital TCP/IP Services for OpenVMS services.

Read QIO error when reading from a socket.

Explanation: Error. An error occurred when reading from a TCP/IP socket.

User Action: Examine the error to determine the cause of the failure and take action based on the error found. See *Digital TCP/IP Services for OpenVMS System Services and C Socket Programming* for information on Digital TCP/IP Services for OpenVMS services.

Rejected function code !UB, unsupported function.

Explanation: Error. A modbus function other than "READ HOLDING REGISTERS", function code "3", or "PRESET MULTIPLE REGISTERS", function code "16", was received by the DAS. These two modbus functions are the only two functions supported by the DAS.

User Action: Reprogram the programmable controller to send a supported modbus function code.

Matching transaction not found for invocation id !UW (timed out).

Explanation: Error. A response was received from the device, but a request with a matching transaction number (invocaton ID) was not found. The most likely cause for this error is that the request timed out and was deleted.

User Action: Increase the I/O timeout (on the path definition) or analyze the network to identify and eliminate bottlenecks in order to improve network response times.

No unsolicited response block available for inv id !UW - discarded.

Explanation: Error. A request from the device was received, but the queue of unsolicited I/O blocks has been exhausted.

User Action: Increase the size of the queue by increasing the value of the parameter ILAN$MOD984_UNSOL_QUEUE_SIZE. See Section 2.3.4.2 for more information on this parameter.

Network address !AD connection lost.

Explanation: Error. The DAS received a connection failed, or timeout status from the network. The DAS starts aborting the connection to the device.

User Action: Determine the cause of the network disconnection and then reconnect to the device.

Network address !AD connection established.

Explanation: Informational. The DAS has successfully established a TCP/IP connection to the device.

User Action: None.

Network address !AD disconnected.

Explanation: Informational. The DAS has successefully disconnected the TCP/IP connection to the device.

User Action: None.

B.4 PE Logged Messages

The following messages are logged by one or more of the Protocol Emulators (PEs).

Unsolicited ID !AD too big. Maximum allowed is !AD.

Explanation: Error. The user attempted to create a physical point with an unsolicited ID that was larger than the current maximum allowed.

User Action: Either increase the maxium unsolicited ID allowed or use an unsolicited ID that is within range. See Section 2.3.4.2 for information on modifying the maximum allowed unsolicited ID value.

Error calling BCC$COMPILE_POINT for unsolicited ID !AD.
Logical point !AD.

Explanation: Error. When defining an unsolicited read point, the logical point is compiled to provide faster access to the logical point and the compile point call failed.

User Action: Examine the error message returned from the BCC$COMPILE_POINT call and take action based on the message found.

Trigger point already defined for unsolicited ID !AD.

Explanation: Error. The user is attempting to define a trigger unsolicited phypoint and a trigger unsolicited phypoint already exists for this unsolicited ID. Duplicate trigger unsolicited phypoints are not allowed.

User Action: None.

Point already defined for unsolicited ID !AD.

Explanation: Error. The user is attempting to define an unsolicited phypoint and an unsolicited phypoint already exists for this unsolicited ID. Duplicate unsolicited phypoints are not allowed.

User Action: None.

Download failed. Controller is in run mode.

Explanation: Error. An attempt to download a logic file to the programmable controller failed because the device is in run mode (not stopped).

User Action: Stop the controller. Then download the logic file to the programmable controller.

Download failed. Memory protect keyswitch is on.

Explanation: Error. An attempt to download a logic file to the programmable controller failed because the memory protect switch on the device is turned on.

User Action: Switch off the memory protect switch. Then download the logic file to the programmable controller.

Port !UB is already logged in.

Explanation: Error. The request failed because another user is already logged into the programmable controller at the indicated port.

User Action: Wait for the other user to log out and then retry the request.

Could not set login port for port !UB.

Explanation: Informational. When downloading a file to the programmable controller, the port currently logged into is greater than three. The DAS normally sets its own address as the device logged into the port, but if the port number is greater than three then the DAS cannot do this. In this case, this message is logged and the download continues.

User Action: None.

Device !AD download started.

Explanation: Informational. A download of a logic file to the programmable controller has been started.

User Action: None.

Device !AD download failed.

Explanation: Error. The download of a logic file to the programmable controller failed.

Explanation: Examine the history file for the reason for the failure and take corrective action based on the error found.

Device !AD download completed.

Explanation: Informational. The download of a logic file to the programmable controller has succeeded.

User Action: None.

Device !AD download cancelled.

Explanation: Error. The download of a logic file to the programmable controller was cancelled before it completed due to length of time to do the download.

User Action: Increase the timeout on the device definition to longer than the longest expected download time.

Device !AD upload started.

Explanation: Informational. An upload of a logic file from the programmable controller has been started.

User Action: None.

Device !AD upload failed.

Explanation: Error. The upload of a logic file from the programmable controller failed.

Explanation: Examine the history file for the reason for the failure and take corrective action based on the error found.

Device !AD upload completed.

Explanation: Informational. The upload of a logic file from the programmable controller has succeeded.

User Action: None.

Device !AD upload cancelled.

Explanation: Error. The upload of a logic file from the programmable controller was cancelled before it completed due to length of time to do the upload.

User Action: Increase the timeout on the device definition to longer than the longest expected upload time.

Error accessing file !AD.

Explanation: Error. An error occurred when accessing the OpenVMS file when performing an upload from or download to the programmable controller.

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

Expected diagnostic data A55A (HEX), received data !XB!XB (HEX).

Explanation: Error. When doing a loop device command, the data received back from the programmable controller did not match the data sent. This error indicates a problem with being able to send and receive packets over the modbus network.

User Action: Check cabling for breaks, shorts, improper length, etc. If communicating using a terminal server, check the port counters for errors. Ensure that baud rate and parity are set properly.

Device error: Illegal function.

Explanation: Error. This error is being returned by the programmable controller.

User Action: Check for another user downloading a logic program to this device.

Device error: Illegal data address.

Explanation: Error. This error is being returned by the programmable controller.

User Action: Check that the address specified in the command is within the range of addresses available in the programmable controller.

Device error: Illegal data value.

Explanation: Error. This error is being returned by the programmable controller.

User Action: Check that the address specified in the command is an address type supported by the programmable controller.

Device error: Failure in associated device.

Explanation: Error. This error is being returned by the programmable controller.

User Action: Replace the programmable controller.

Device error: Device busy.

Explanation: Error. This error is being returned by the programmable controller.

User Action: Try the operation again later.

Device error: Negative acknowledgement.

Explanation: Error. This error is being returned by the programmable controller.

User Action: Retry the operation.

Device error: Not logged in.

Explanation: Error. This error is being returned by the programmable controller.

User Action: Check for other users executing control operations (such as start or stop) while a logic program is being downloaded to the programmable controller.

Device error: Illegal configuration.

Explanation: Error. This error is being returned by the programmable controller.

User Action: Reconfigure the programmable controller configuration table.

No logical point defined for unsolicited ID !AD.

Explanation: Error. A read request was received from the programmable controller, but a logical point does not exist for the physical point with the requested unsolicited ID.

User Action: Modify the physical point to put a valid logical point name in the unsolicited ID. See Unsolicited IDs for Read for details on formating of the unsolicited ID.

Too little data for unsolicited ID !AD.
Requested !UL words, but only !UL words in logical point.

Explanation: Error. The size of the logical point is smaller than the size of the data that the programmable controller requested. The data from the logical point is sent and the remainder of the data requested by the programmable controller is set to zeroes.

User Action: Modify the programmable controller program to request a data size that is less than or equal to the size of the corresponding logical point or modify the logical point to be larger in size.

Error calling BCC$GET_VALUE for unsolicited ID !AD.

Explanation: Error. A request to read data from the host was received and a matching physical point was found, but the call to get the current value of the logical point failed.

User Action: Examine the error message returned and take action based on the error found. If the logical point exists then it may be necessary to force recompiling of the logical point name by stopping BASEstar software, deleting the ILAN$_SECTION.DAT file and restarting BASEstar software.

Error in point status for unsolicited ID !AD.

Explanation: Error. A request to read data from the host was received, a matching physical point was found and the call to BCC$GET_VALUE succeeded, but the status of the logical point is bad.

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

Request for unsolicited ID !AD exceeds maximum allowed of !AD.

Explanation: Error. An unsolicited request (read or write) was received, but the address in the request is not within the range currently allowed by the DAS.

User Action: Either increase the maxium unsolicited ID allowed and create a matching physical point or change the programmable controller logic to us an unsolicited ID that is within range. See Section 2.3.4.2 for information on modifying the maximum allowed unsolicited ID value.

Request for unsolicited ID !AD, unsolicited point is not configured.

Explanation: Error. An unsolicited request (read or write) was received, but no matching physical point exists for the address received in the message.

User Action: Create a physical point with an unsolicited ID that matches the address in the message or modify the programmable controller to use an address of an existing physical point.

No license is active for this software product.

Explanation: Error. The DAS could not find a license for the product.

User Action: Register and load a valid license for this DAS.

Index

A

Accessing DAS functions, 3–1
Addressing
  Modicon 584, 3–4
  Modicon 984, 3–4
  Modicon Quantum, 3–4
Automatic data collection, 3–6
  pollsets, 3–6
  unsolicited, 3–6

C

Configuration file
  device records, 2–9
  editing, 2–6
  path records
    line parameters, 2–6
  type records, 2–6
Configuring parameters
  ILAN$MAX_SPT_REQUESTS, 2–10
  ILAN$MOD984_MAX_REG_ADDRESS, 2–13
  ILAN$MOD984_UNSOL_QUEUE_SIZE, 2–13
  ILAN$MODBUS_TRANSMIT_DELAY, 2–12
  ILAN$MODHOST_MAX_REG_ADDRESS, 2–13
  ILAN$MODHOST_UNSOL_QUEUE_SIZE, 2–13
  ILAN$MODHOST_UNSOL_WRITE, 2–13
  ILAN$MODTCP_DROP_TIME, 2–13
  ILAN$MODTCP_PROBE_TIME, 2–13
Configuring ports, 2–11

D

Data type formats
  Modicon 584, 3–5
  Modicon 984, 3–5
  Modicon Quantum, 3–5
  structured data, 3–3
Device access software
  accessing functions, 3–1
  description of, 1–1
  installation of, 2–1
  installation requirements, 2–1
  overview, 1–1
  supported devices, 1–2
Device access software (cont'd)
  supported functions, 1–2, 3–2
  using, 3–1
Device communications
  tracing, 2–14
Device records, 2–9
Devices, 1–1
  communications, 1–1
  Modicon, 1–2
Disk space requirements, 2–2
Download function, 3–2

F

Files
  created during installation, 2–4
Functions
  read data, 3–3
  write data, 3–3

H

Hardware
  required, 2–1

I

Installation, 2–1
  files created, 2–4
  messages, 2–5
  postinstallation tasks, 2–6
  procedure, 2–2
  requirements, 2–1

L

Line parameters, 2–6

M

Messages
  installation, 2–5
  logged, B–1
Modbus
  network address, 2–9
Modbus Plus
  network address
    host, 2–9

Modbus Plus
network address (cont'd)
slave, 2–9
Modicon 584
addressing, 3–4
data type formats, 3–5
read data, 3–4
write data, 3–4
Modicon 984
addressing, 3–4
data type formats, 3–5
read data, 3–4
write data, 3–4
Modicon Quantum
addressing, 3–4
data type formats, 3–5
read data, 3–4
write data, 3–4

P

Path records, 2–6
Plant-floor equipment
setting up, 2–14
Postinstallation tasks, 2–6
Problem reporting, 2–15
Product failure, 2–15

R

Read data function
Modicon 584, 3–4
Modicon 984, 3–4
Modicon Quantum, 3–4
structured data, 3–1
Read data functions, 3–3
Read status function, 3–5

S

Software
additional required, 2–2
required, 2–2
SPT block usage, 2–10
Start function, 3–2
Stop function, 3–2
Structured data, 3–1
mixed data types, 3–1

T

TCP/IP
network address, 2–10
setting parameters, 2–13
Trace
using, 2–14
Type records, 2–6

U

Unsolicited
setting parameters, 2–13
Unsolicited counters, 3–6
resetting, 3–6
Unsolicited ID
read, 3–7
write, 3–7
Unsolicted ID
syntax, 3–7
Upload function, 3–2

W

Write data function
Modicon 584, 3–4
Modicon 984, 3–4
Modicon Quantum, 3–4
structured data, 3–1
Write data functions, 3–3