BASEstar™ Classic DAS for Siemens H1™ Protocol

Installation and User's Guide

Order Number: AA-R219B-TE

April 2000

This manual describes how to install and use the DAS for Siemens H1 Protocol 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 Siemens H1 Protocol, Version 3.4A

Compaq Computer Corporation
Houston, Texas

April 2000

© 1991 Compaq Computer Corporation.

Compaq, VMS and VAX 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.

H1, Simatic, and SINEC are trademarks of Siemens AG.

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 v

1 Overview

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

2 Installing the DAS

2.1 Installation Requirements 2–1
2.1.1 Hardware Requirements 2–1
2.1.2 Software Requirements 2–2
2.1.3 Disk Space Requirements 2–2
2.2 Installation 2–3
2.2.1 Files Created During Installation 2–5
2.2.2 Installation Messages 2–5
2.3 Post Installation Tasks 2–6
2.3.1 Editing the Configuration File 2–7
2.3.1.1 Editing Type Records 2–7
2.3.1.2 Editing Path Records 2–7
2.3.1.3 Editing Device Records 2–8
2.3.2 Configuring DECnet Plus Parameters 2–10
2.3.3 DAS SPT Block Usage 2–12
2.3.4 Setting DAS-Specific Parameters 2–13
2.3.5 Setting Up Plant Floor Equipment 2–14
2.4 Tracing OSI Messages 2–14
2.5 Failures During Product Use 2–14

3 Using the DAS

3.1 Accessing DAS Functions 3–1
3.2 Supported Functions 3–1
3.2.1 Supported Communications Cards 3–2
3.2.2 TSAPs, Establishing Communications 3–3
3.2.3 Control Operations 3–4
3.2.3.1 Control Operation Communications 3–4
3.2.3.2 Model S135U Communications 3–4
3.2.3.3 Common Steps 3–4
3.2.3.4 Start and Stop 3–5
3.2.3.5 Upload and Download 3–5
3.2.3.6 Read Status 3–6

3.2.4 Read and Write Data 3–8
3.2.4.1 Data Block Access 3–8
3.2.4.2 Read and Write Message 3–10
3.2.5 Send and Receive Message 3–13
3.3 Automatic Data Collection 3–13
3.3.1 Unsolicited Data 3–13
3.3.2 Pollsets 3–14

A Logged Messages

A.1 NI Logged Messages A–1
A.2 PE Logged Messages A–7

B Sending and Receiving Data Using ILAN$DEVICE_SPECIFIC

C Data exchange using the unsolicited TSAP

C.1 Read Operation C–1
C.2 Acknowledged Write Operation C–2
C.3 Non Acknowledged Write Operation C–3
C.4 Unsolicited messages C–3

Index

Examples

2–1 NULL Internet OSI Transport Template 2–11
2–2 OSI Transport 2–11

Figures

1–1 DAS Communications 1–2

Tables

1–1 Siemens Devices and Supported Functions 1–3
2–1 Disk Space Requirements 2–3
2–2 Files Created During Installation 2–5
2–3 Path Parameters 2–8
2–4 OSI Transport Template Parameters 2–10
2–5 OSI Transport Parameters 2–10
2–6 SPT Static Block Sizes 2–13
2–7 SPT Dynamic Block Sizes 2–13
2–8 TI Parameters 2–14
3–1 Siemens Devices and Supported Functions 3–2
3–2 Data Block Addressing 3–9
3–3 TI Addressing 3–10

Preface

This document describes how to install and use the BASEstar Classic DAS for Siemens H1 Protocol.

Intended Audience

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

- BASEstar Classic for OpenVMS software
- BASEstar Classic DAS for Siemens H1 Protocol software

This document is also intended for use by 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 Seimens Siemens 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 Siemens H1 Protocol.
- Chapter 2 provides information you need to install the DAS for Siemens H1 Protocol.
- Chapter 3 provides information about the supported functions for Siemens programmable controllers, and how to access the functions.

Associated Documents

Further information on topics covered in this document 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*

Information on DECnet Plus installation, configuration and programming can be found in the following documents.

- *DECnet Plus for OpenVMS Installation and Configuration*
- *DECnet Plus for OpenVMS Introduction and Planning*
- *DECnet Plus for OpenVMS Network Management*
- *DECnet Plus for OpenVMS Common Trace Facility Use*
- *DECnet Plus for OpenVMS Programming*

Refer also the documentation produced by Siemens for the SH1 Type S115U, SH1 Type S135U, SH1 Type S150U and SH1 Type S150U programmable controllers.

Conventions

This manual 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 Siemens H1 Protocol. It also includes a brief description of Siemens devices and the supported functions for the DAS.

1.1 Description

The DAS for Siemens H1 Protocol allows you to access Siemens devices using the H1 protocol through BASEstar Classic device connection management. Device connection management is a component of BASEstar Classic software, which is an engineered family of software components and services that facilitate the development and integration of manufacturing automation solutions.

Using the DAS for Siemens H1 Protocol, users or applications can perform a variety of device access functions for the Siemens family of devices using the H1 protocol. Examples of functions supported for the Siemens devices include: reading and writing data, uploading and downloading logic programs, and reading the status of a device.

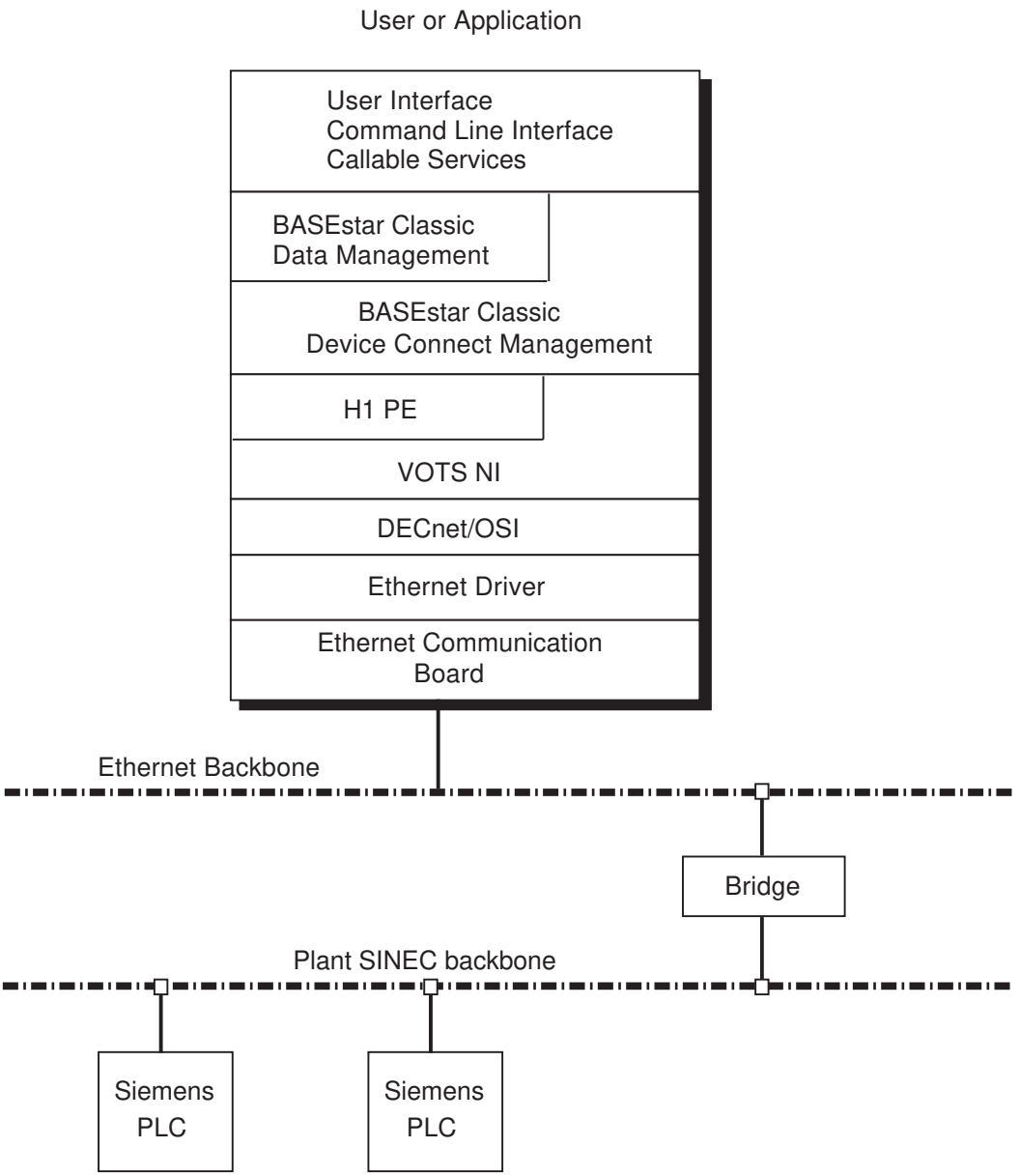
1.2 Device Communications

The DAS for Siemens H1 Protocol consists of a Protocol Emulator (PE) and a Network Interface (NI). PEs and NIs work together to provide device-specific communications for BASEstar Classic's generic callable services.

The PE translates BASEstar Classic device connection management generic services into device-understandable format. The PE also converts device-specific protocol into BASEstar Classic device connection management format. The NI works directly with an 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 device connection management and the device.

The DAS for Siemens H1 Protocol must be installed on an OpenVMS host that is connected to Siemens programmable controllers using an Ethernet backbone. Optionally, one may introduce an Ethernet bridge to separate the Ethernet traffic in the computer room from the Ethernet traffic on the plant floor. The OpenVMS host and the programmable controller communicate using the first four layers of the ISO/OSI standard. On the OpenVMS side this is done using the OSI Transport Services application programming interface.

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 library management and BASEstar Classic device connection management menu system, commands, and callable services.

The DAS for Siemens H1 Protocol 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 a OpenVMS file
- Download a 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 for a device

The DAS for Siemens H1 Protocol supports Siemens programmable controllers model S115U, S135U, S150U and S155U.

The S135U programmable controllers may be configured with a single type "S" or type "R" CPU. This programmable controller must include at least one CP-535 Ethernet communications card. For model S150U programmable controllers it is recommended that a CP-511 serial communications card is also used.

The DAS for Siemens H1 Protocol supports a variety of devices, as shown in Table 1–1.

Table 1–1 Siemens Devices and Supported Functions

Device	Upload	Download	Start	Stop	Read Data	Write Data	Read Status
S115U	X	X	X	X	X	X	X
S135U	X	X	X	X	X	X	X
S150U	X	X	X	X	X	X	X
S155U					X	X	
COMPATIBLE[1]					X	X	

[1]The 'COMPATIBLE' type device is intended to support TI devices with an H1 ethernet interface.

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

2
Installing the DAS

This chapter provides the information you need to install the DAS for Siemens H1 Protocol and to configure your system.

2.1 Installation Requirements

The DAS for Siemens H1 Protocol can be installed by a system manager who has access to an account with the SYSPRV privilege. However, modifications to the hardware configuration of the host must be done by trained Digital service personnel.

The installation takes 1 to 5 minutes depending on distribution media and system configuration, and is carried out using the standard OpenVMS utility VMSINSTAL. You only need to respond to simple YES or NO questions during the installation.

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 Requirements

The DAS for Siemens H1 Protocol communicates with Siemens programmable controllers using the H1 protocol over Ethernet. The DAS for Siemens H1 Protocol requires the same hardware as BASEstar Classic device connection management. For more information on hardware requirements, see the *BASEstar Classic Installation Guide*. The same Ethernet controller may be shared between DECnet Plus and the DAS for Siemens H1 Protocol.

Siemens model S135 programmable controllers are multiprocessor programmable controllers. They allow multiple CPUs to coexist in the same programmable controller. There is one MUX/Coordinator card which coordinates data traffic between the CPUs and to the Ethernet communication card. When performing control operations such as upload or download, it is necessary to connect a special RS-232 cable between the Ethernet communications card and the CPUs or between the Ethernet communications card and the Coordinator.

If the RS-232 cable is connected to the Coordinator card, then control operations may be performed to any one of the CPUs.

Note

It is recommended that the DIP switch on the Coordinator card is set so that the leftmost CPU is addressed as CPU number 1. Also, the toggle switch of the Coordinator should be set to RUN or TEST mode.

If the RS-232 cable is connected directly to the CPU card, then all control operations are restricted to that CPU only.

2.1.2 Software Requirements

Before installing the DAS for Siemens H1 Protocol, the following software must already be installed:

- OpenVMS Version 5.5-2 or higher (VAX)
- OpenVMS Version 6.1 or higher (Alpha)
- VAX/FMS Version 2.3 or Version 2.4 (required only when the menu system will be used) (The Menu System is available only on OpenVMS/VAX systems.)
- BASEstar Classic for OpenVMS, Version 3.4 or higher
- DECnet Plus, Version 5.6B or higher (VAX)
- DECnet Plus, Version 5.7 or higher (Alpha)

For information on installing the above software, see the *BASEstar Classic Installation Guide* and *DECnet Plus for OpenVMS Installation and Configuration*.

The minimum required version of DECnet Plus is different for the various versions of OpenVMS. Refer to *DECnet Plus for OpenVMS Installation and Configuration* for the minimum required version of OpenVMS required for the version of DECnet Plus being installed.

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 Disk Space Requirements

Table 2–1 lists the disk space required to install the DAS for Siemens H1 Protocol. The space requirements are approximations; actual sizes may vary depending on your system environment, configuration, and software options selected.

Table 2–1 Disk Space Requirements

Peak/Net Usage	Approximate Space Requirements
Peak usage (during installation)	500 blocks (VAX)
	600 blocks (Alpha)
Net usage (after installation)	300 blocks (VAX)
	400 blocks (Alpha)

2.2 Installation

When your system meets all the hardware and software requirements, you can install the DAS for Siemens H1 Protocol. The installation takes from 1 to 5 minutes, depending on system load and configuration.

Install the DAS for Siemens H1 Protocol 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 DCM_H1VVA034 ddcu:
```

   The **DCM_H1VVA034** 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 does not need to be running in order 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 29-FEB-2000 at 14:16.

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_H1VVA V3.4

        Beginning installation of DCM_H1VVA V3.4 at 14:17

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

   Copyright 1991 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 Siemens H1 Protocol installation procedure.

   Checking for a valid license...


        Product:      DAS-SH1-CL
        Producer:     DEC
        Version:      3.4
        Release Date: 13-DEC-1996

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

   Now checking OpenVMS version...

   Now checking that network software is installed...

   Now checking that BASEstar Classic is installed...

   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]?

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

   The configuration template file for Siemens H1 support,
   DCM_H1_CONFIG.TEMPLATE, is used to define the Siemens 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.

%VMSINSTAL-I-SYSDIR, This product creates system directory [SYSHLP.EXAMPLES.DCM_
H1].

   The BASEstar Classic parameter ILAN$H1_TI_CR_BASE has just been created.

Parameter ILAN$H1_TI_CR_BASE, type String.
    Current value is "",  Default value is "".

   The BASEstar Classic parameter ILAN$H1_TI_CR_SIZE has just been created.

Parameter ILAN$H1_TI_CR_SIZE, type Value.
    Current value is 1,  Default value is 1.
    Minimum value is 1,  Maximum value is 10000.

   The BASEstar Classic parameter ILAN$H1_TI_V_BASE has just been created.

Parameter ILAN$H1_TI_V_BASE, type String.
    Current value is "",  Default value is "".

   The BASEstar Classic parameter ILAN$H1_TI_V_SIZE has just been created.

Parameter ILAN$H1_TI_V_SIZE, type Value.
    Current value is 1,  Default value is 1.
    Minimum value is 1,  Maximum value is 10000.
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...

   Copyright 1991 Compaq Computer Corporation

   Confidential commercial computer software.  Valid license required.
```

```
Executing the Installation Verification Procedure.

BASEstar Classic DAS for Siemens H1 Protocol installation procedure
has succeeded.

     Installation of DCM_H1VVA V3.4 completed at 14:18

     VMSINSTAL procedure done at 14:18
```

2.2.1 Files Created During Installation

Table 2–2 lists the files that the DAS for Siemens H1 Protocol installation procedure creates and the directories in which they are placed.

Table 2–2 Files Created During Installation

Directory	Filename
BCC$SYSDATA:	DCM_H1_CONFIG.TEMPLATE
SYS$LIBRARY:	ILAN_SH1.EXE
	ILAN_VOTS.EXE
	DCM_H1$DEF.H
SYS$HELP:	DCM_H1VVA034.RELEASE_NOTES (VAX)
	DCM_H1VAA034.RELEASE_NOTES (Alpha)
SYS$TEST:	DCM_H1$IVP.COM
SYS$COMMON:[SYSHLP .EXAMPLES.DCM_H1]	DCM_H1_SEND_EXAMPLE.C
	DCM_H1_RECEIVE_EXAMPLE.C

2.2.2 Installation Messages

You may see VMSINSTAL messages during the installation procedure. The following messages are specific to the DAS for Siemens H1 Protocol installation:

BADBCC, BASEstar Classic software must be installed before DAS for Siemens H1 Protocol.

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 Siemens H1 Protocol.

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.

BADDECNET, A valid version of DECnet must be installed before the DAS for Siemens H1 Protocol.

Explanation: Error. Incorrect version of DECnet Plus software.

User Action: Install DECnet Plus software Version 5.6B (VAX) or Version 5.7 (Alpha) or higher.

BADVMS (VAX), The DAS for Siemens H1 Protocol 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 Siemens H1 Protocol 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), DAS for Siemens H1 Protocol requires 300 blocks after installation.

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

User Action: Delete any unnecessary files, then reinstall.

NETBLOCKS (Alpha), DAS for Siemens H1 Protocol requires 400 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 Post Installation Tasks

This section describes the tasks to perform after installing the DAS for Siemens H1 Protocol, including editing the configuration file, setting the device connection management support block parameter, configuring ports and setting up plant floor equipment.

2.3.1 Editing the Configuration File

A configuration file, BCC$SYSDATA:DCM_H1_CONFIG.TEMPLATE, is supplied with the 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 this file to include site-specific information about types and paths, and execute the file.

The following sections give examples of the type, path, and device records. See 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 SH1_TYPE_S115U /manuf = SIEMENS /model = S115U-
   /proto = SH1 /descr = SH1_CPU_S115U

create type SH1_TYPE_S135U /manuf = SIEMENS /model = S135U-
   /proto = SH1 /descr = SH1_CPU_S135U

create type SH1_TYPE_S150U /manuf = SIEMENS /model = S150U-
   /proto = SH1 /descr = SH1_CPU_S150U

create type SH1_TYPE_S155U /manuf = SIEMENS /model = S155U-
   /proto = SH1 /descr = SH1_CPU_S155U

create type SH1_COMPATIBLE /manuf = UNKNOWN /model = H1_COMPATIBLE-
   /proto = SH1 /descr = "Any device supporting H1 protocol on Ethernet"
```

The model field is used to tell the DAS the model of the Siemens programmable controller being configured. The DAS uses this field to determine what functions are not supported (for S155U programmable controllers), to check if start/stop options should be checked (for S135U programmable controllers) and to customize commands sent to the programmable controller. Be sure that the model field matches the model of the programmable controller being configured.

The `COMPATIBLE` type supports read and write functions only and can be used to configure TI devices that support the Siemens H1 protocol.

2.3.1.2 Editing Path Records

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

```
create path OSI_PATH /netn = VOTS /multidrop-
   /vaxport = "_OS:" /io_size = 1024 /retries = 0 /descr = OSI_PATH
```

Table 2–3 lists the path parameters and indicates the value(s) that are allowed for each.

Table 2–3 Path Parameters

Parameter	Value(s)
VAXport[1]	_OS:
Netname	VOTS
Multidrop[2]	MULTIDROP
Timeout[3]	1-59
Retries[4]	
IO Size[5]	

[1]This parameter defines the pseudo-device used to connect to the OSI network.

[2]This parameter must be MULTIDROP since all H1 devices will use the same VAXport.

[3]This parameter specifies in seconds how long it takes to time-out an I/O request. If this parameter is not specified or is zero, then a default value of 9 seconds is used.

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

[5]The parameter is used to set the maximum I/O size for reads and writes to the programmable controller. See the documentation for your device to set this parameter. The I/O size includes the size of the header sent in requests to the device. There is no default value for this parameter.

The line parameters in the path definition are not used by this DAS and are ignored.

2.3.1.3 Editing Device Records

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

```
create device S150_DEVICE /type = SH1_TYPE_S150U / path = OSI_PATH -
   /neta = "H1NET%080006010001:(R=READREAD,W=WRITWRIT)"
```

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 longer than the timeout on the path definition. To set the actual I/O timeout use the timeout on the path definition.

The timeout for a connect is controlled only by the CR Timeout on the OSI transport template definition. See Section 2.3.2 for information on setting OSI transport template parameters.

The device may be defined as "unsolicited". See Section 3.3.1 for information on unsolicited data collection.

Device address format

When configuring the Siemens device, the network address qualifier must reflect the address of the programmable controller.

This network address must at least contain the OSI transport template name used by DECnet Plus and the Ethernet address of the device.

It has the following format:

osi_template_name%Ethernet_address:

For example:

H1NET%080006000001:

Note that the OSI transport template name and Ethernet address are separated by " **%** " and that the Ethernet address must be terminated by " **:** ".

Optionally, up to five Transport Service Access Points (TSAPs) may be added to define the read, write, unsolicited or command channels and to define the calling TSAP.

- **R** defines the TSAP for the read channel. It has no default value. This channel is used for read operations.
- **W** defines the TSAP for the write channel. It has no default value. This channel is used for write operations.
- **U** defines the TSAP for the unsolicited channel. It has no default value. This channel is used to send and receive messages.
- **C** defines the TSAP for the command channel. Its default value is **S5PGCONN**. This channel is used for command operations (upload, download, stop, start,...)
- **O** defines the calling TSAP on the host. Its default value is **VAXVAXVA**.

For example:

```
H1NET%080006000001:(R=READREAD)
H1NET%080006000001:(U=UNSOLUNS)
H1NET%080006000001:(R=READTSAP,W=WRITTSAP)
H1NET%080006000001:(R=MY_TSAP1,W=MY_TSAP2,U=MY_TSAP3)
```

These TSAPs must also be defined on the CP-535 card of the programmable controller. The names of the TSAPs must be up to 8 uppercase characters, left justified, blanks padded on the right. They are enclosed in parenthesis and separated by commas.

Temporary TSAPs are accessed when the command operation starts and are deaccessed when the operation is completed. Permanent TSAPs are accessed when the device is enabled (if the device is "unsolicited") or at the first requested operation (if the device not "unsolicited"). Permanent TSAPs are deaccessed when the device is disabled.

The unsolicited TSAP is always a permanent TSAP.

The command TSAP is always a temporary TSAP.

By default, the read TSAP and the write TSAP are permanent TSAPs. It is possible to make them temporary, by defining them with **TR** and **TW** instead of **R** and **W**.

For example:

```
H1NET%080006000001:(TR=TEMPREAD)
H1NET%080006000001:(R=SIEMENSR,TW=SIEMENSW)
H1NET%080006000001:(TR=TMP_READ,TW=TMP_WRIT)
```

The network address is limited to 55 characters. If the OSI transport template length + ethernet address length + TSAP lengths is larger than 55 characters then additional TSAPs may be defined as described below:

Additional TSAPs may be defined using a system logical name. The name of this logical consists of the OSI transport template name used by DECnet Plus and the Ethernet address of the device, separated by a **$** .

For example:

```
DEFINE/SYSTEM H1NET$080006000001 "(C=CDE_TSAP,O=OWN_TSAP)"
```

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

2.3.2 Configuring DECnet Plus Parameters

Use SYS$STARTUP:NET$CONFIGURE to configure your OSI Transport Template for Siemens communications.

Note

Be sure to configure the template as a NULL Internet transport template.

Table 2–4 gives a list of recommended OSI transport template parameters. Table 2–5 gives a list of OSI transport parameters that may need to be modified for your intallation. These parameters are explained in detail in the DECnet Plus documentation. Some are inter-dependent so that modifying one parameter may require changing other ones also.

Table 2–4 OSI Transport Template Parameters

Parameter	Recommended Value[1]
Keepalive Time	3[2]
Retransmit Threshold	2[2]
CR Timeout	5
ER Timeout	5

[1]The values of these parameters depends on your specific network configuration and expected response times.

[2]The product of these values must be lower than the /TIMEOUT value on the device definition.

Table 2–5 OSI Transport Parameters

Parameter
Maximum Transport Connections
Maximum Remote NSAPs

Other timers in the transport template may need to be modified depending on your specific hardware and network configuration.

To modify transport template values place an NCL script in SYS$COMMON:[SYSMGR] and execute it from your system startup command file. Use the following syntax to execute an NCL script.

```
$RUN SYS$SYSTEM:NCL
@SYS$MANAGER:CUSTOM_NCL_SCRIPT.NCL
```

Example 2–1 shows an example OSI transport template configured for NULL Internet.

Example 2–1 NULL Internet OSI Transport Template

```
NCL>sho osi transport template h1net all

Node 0 OSI Transport Template h1net
at 1996-10-17-14:49:41.446-04:00Iinf

Identifiers

    Name                              = h1net

Characteristics

    Keepalive Time                    = 60
    Retransmit Threshold              = 8
    Initial Retransmit Time           = 5
    CR Timeout                        = 30
    ER Timeout                        = 30
    Network Service                   = CLNS
    Security                          = <Default value>
    Classes                           =
       {
          4
       }
    Checksums                         = True
    Maximum NSDU Size                 = 2048
    Expedited Data                    = True
    CONS Template                     = <Default value>
    Use CLNS Error Reports            = True
    Acknowledgement Delay Time        = 1
    Local NSAP                        = <Default value>
    CLNS Inactive Area Address        =
       {
          49::FF-00
       }
    Inbound                           = True
    Loopback                          = False
    Send Implementation Id            = True
    Extended Format                   = True
    Network Priority                  = 0
    Send Preferred maximum TPDU size  = True
    Send Request Acknowledgement      = True
    RFC1006 Port Number               = 102
```

Example 2–2 shows an example OSI transport definition.

Example 2–2 OSI Transport

```
NCL>sho osi transport all char

Node 0 OSI Transport
at 1996-10-17-14:57:47.388-04:00Iinf

Characteristics
```

(continued on next page)

Example 2–2 (Cont.) OSI Transport

```
Maximum Transport Connections     = 200
Maximum Receive Buffers           = 4000
Delay Weight                      = 5
Delay Factor                      = 4
Maximum Window                    = 20
Maximum Multiplexing              = 4294967295
Disconnect Holdback               = 0
Maximum Network Connections       = 4294967295
ISO Version                       = 1
Version                           = V1.1.0
CONS Classes Supported            =
   {
      0 ,
      2 ,
      4
   }
CLNS Classes Supported            =
   {
      4
   }
Support MAP                       = False
Maximum Remote NSAPs              = 201
NSAP Selector                     = 33
CONS Filters                      =
   {
      "OSI Transport"
   }
Congestion Avoidance              = False
CONS NSAP Addresses               =
   {
   }
Maximum RFC1006 Connections       = 4294967295
RFC1006 Listener Ports            =
   {
       102
   }
Out Of Order Cache Size           = 10
```

Furthermore there is a similar set of parameters pre-defined for the Siemens CP-535 communications card. The parameters on the host must match the pre-defined parameters on the Siemens CP-535. Setting the parameters at the recommended values will allow normal communications with the Siemens programmable controller. It is possible to choose other values for these parameters depending on your special network requirements. However, please consult DECnet Plus documentation carefully before changing any of these parameters.

2.3.3 DAS SPT Block Usage

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 2–6 shows the static blocks of each size that are used by the DAS.

Table 2–6 SPT Static Block Sizes

Block Size	Quantity[1]	Block Type
MEDIUM	2	Device
SMALL	1	Device[2]
	1	Unsolicited point[2]

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

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

Table 2–7 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.

Table 2–7 SPT Dynamic Block Sizes

Block Size	Quantity[1]	I/O Type
LARGE	1	Upload/download
	1	Read status
MEDIUM	1	Read/write[2]
	1	Start/Stop[2]
	1	Send/Receive message
SMALL	1	Any

[1]Quantity is quantity per I/O.

[2]The block size depends on the size of the IO_SIZE on the path definition. If the IO_SIZE is larger than about 400, the block size will be a LARGE block.

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.

2.3.4 Setting DAS-Specific Parameters

None of the parameters created for the DAS for Siemens H1 Protocol are dynamic. A device must be disabled and reenabled before the parameter modifications become effective.

The installation procedure for the DAS for Siemens H1 Protocol creates BASEstar Classic parameters to allow you to tune the DAS environment for accessing data in TI programmable controllers. These parameters are listed in Table 2–8.

Table 2–8 TI Parameters

Parameter	Description
ILAN$H1_TI_CR_BASE	This value specifies the starting block number for the control registers in the programmable controller. The value is specified as a "DB:XX" value. There is no default value for this parameter.
ILAN$H1_TI_V_BASE	This value specifies the starting block number for the variable memory in the programmable controller. The value is specified as a "DB:XX" value. There is no default value for this parameter.
ILAN$H1_TI_CR_SIZE	This value specifies the size of the DB block for control registers. The value is used to calculate which DB block to request when reading a CRXXXXX address. There is no default value for this parameter.
ILAN$H1_TI_V_SIZE	This value specifies the size of the DB block for variable memory. The value is used to calculate which DB block to request when reading a VXXXXX address. There is no default value for this parameter.

See Addressing Syntax for TI Devices for information on addressing for TI programmable controllers and how TI addresses are translated into Siemens H1 addresses.

2.3.5 Setting Up Plant Floor Equipment

To set up your plant floor equipment, see the Siemens documentation for your specific device.

2.4 Tracing OSI Messages

The tracing facility enables you to see if, and what actually is send out and received from OpenVMS on the OSI stack.

You could invoke the Common Trace Facility for a full and live trace by typing:

```
$ TRACE
CTF> START/LIVE/FULL "OSITP CR MESSAGES *"
```

The above trace will trace all connect request protocol data units.

See *DECnet Plus for OpenVMS Common Trace Facility Use* for a further description of the Common Trace Facility, tracepoints supported and the output generated.

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 DECnet Plus you are using
  - The version number of BASEstar Classic you are using
  - The hardware system you are using (such as a model number)

  - The Siemens 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 Siemens devices, and how to access these functions.

3.1 Accessing DAS Functions

DAS for Siemens H1 Protocol functions are accessed through the BASEstar Classic device connection management:

- Commands
- Menu system
- Callable services

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

```
$  BSTAR DCM
```

For more information about the BASEstar Classic device connection management commands, refer to the *BASEstar Classic Command Line Interface User's Guide*.

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

```
$  BSTAR/MENU
```

Note

The menu system is available on OpenVMS/VAX systems only.

For information about the BASEstar Classic device connection management 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 Application Programming Interface Reference Guide*.

3.2 Supported Functions

This section describes the functions that are supported by the DAS for Siemens H1 Protocol. It explains how these operations are performed from the point of view of the host and from the point of view of the programmable controller. The more complex control operations such as upload, download, start, stop, status are explained in greater detail outlining as much as possible the steps involved. Table 3–1 shows the Siemens devices and the functions supported for them.

Table 3–1 Siemens Devices and Supported Functions

Device	Upload	Download	Start	Stop	Read Data	Write Data	Read Status
S115U	X	X	X	X	X	X	X
S135U	X	X	X	X	X	X	X
S150U	X	X	X	X	X	X	X
S155U					X	X	
COMPATIBLE[1]					X	X	

[1]The 'COMPATIBLE' type device is intended to support TI devices with an H1 ethernet interface.

The host and the Siemens programmable controller are interconnected using Ethernet. The basic communications are carried using the first four layers of the ISO/OSI communications reference model. On the host side this is done using DECnet Plus which implements the first four layers. DECnet Plus provides the basic communication environment for data exchange. As such it allows for programs such as DAS for Siemens H1 Protocol to establish Transport connections with the Siemens programmable controller, and then to exchange data packets to transfer information. These data packets are encoded according to the Siemens specific protocol called H1. The DAS for Siemens H1 Protocol implements the H1 protocol to send commands to the programmable controller to obtain data, to understand the meaning of the data, and to make it available to application programs on the host.

For upload or download operations, the application program provides the DAS for Siemens H1 Protocol with the name of a file on the host. The DAS for Siemens H1 Protocol then sends the file to the memory of the programmable controller.

3.2.1 Supported Communications Cards

From the point of view of the programmable controller there are several cards which cooperate to execute the operations. First, there is the CPU and the memory card. Next there is the CP-535 Ethernet communications card which connects the programmable controller to the Ethernet backbone.

For normal read or write operations, the CP-535 exchanges data with the CPU using a special dual-port RAM. Normally this dual-port RAM needs to be synchronized between the CP-535 and the CPU. Please refer to the appropriate Siemens documentation for the required programming steps on the CPU of the programmable controller to do this.

For control operations such as upload, download, status, start and stop, the CPU is sometimes stopped and thus the dual-port RAM is inoperable. In this case, a second communication card is used, the CP-511. This is an RS-232 based communication card which is able to communicate directly with the CPU even when the CPU is stopped.

The CP-535 and the CP-511 cards are interconnected using a special cable. The control operation is sent from the host to the CP-535, the CP-535 sends it to the CP-511 which then forwards the operation to the CPU.

For model S135U programmable controllers, the CP-511 card is not required as the CPU card itself may be connected directly to the CP-535.

Model S135U programmable controller allows for multiple CPU cards within the same programmable controller. To coordinate between the CPUs, there is a special Coordinator/MUX card. In this configuration, read and write operations are done as before using the dual-port RAM and data is exchanged with the host as before.

For control operations, the special RS-232 cable must be connected between the CP-535 and any of the CPUs or between the CP-535 and the Coordinator. When the cable is connected directly to the CPU, the programmable controller appears to the host as a uniprocessor programmable controller. When the special RS-232 cable is connected to the Coordinator card, then control operations may be performed to any of the CPUs. Thus it is not necessary to move the cable from one CPU to the other to upload the contents of any CPUs in the programmable controller.

In the case uploads may be performed remotely without having to move the cable between the CP-535 and each of the CPUs in the programmable controller.

3.2.2 TSAPs, Establishing Communications

The transport service provides transport users with the means to set up, transfer data across and conclude transport connections. The transport connection is a two-way data transfer path between two transport users. On the programmable controller side the transport user is the CP-535 communication card. On the host side the transport user is the DAS for Siemens H1 Protocol. Each transport user is associated with a TSAP.

Application programs do not need to have separate TSAPs. They do not establish a transport connection directly to the programmable controller. They all use the transport connection established between the programmable controller and the DAS for Siemens H1 Protocol.

The number of available TSAPs is a limited resource both on the host and especially on the programmable controller. Therefore, all application programs on the same host use only one channel.

The name of the TSAP on the host is fixed by default as **VAXVAXVA** .

The host may connect with up to three TSAPs on the programmable controller. Available TSAPs are the read, write, and unsolicited TSAPs. The read TSAP is used to read data from the programmable controller. The write TSAP is used to write data to the programmable controller. The unsolicited TSAP is used to send or receive data packets.

In addition to these there is a fixed command TSAP on the programmable controller which is used for control operations such as upload, download, start, stop and status.

The name of this command TSAP is fixed by default as **S5PGCONN** .

There can only be one connection at a time to this special TSAP. The programmable controller allows only one user or application program at a time to do upload, download, status, etc.

All the TSAPs names are configurable. On the host this is specified as part of the /NETADDR qualifier of the device definition. See Device address format for further information.

3.2.3 Control Operations

The control operations implemented with DAS for Siemens H1 Protocol are upload, download, start, stop, and status. Only one application program or user may do one of these at any time.

3.2.3.1 Control Operation Communications

For control operations a special TSAP is used. This is predefined and may be changed. When doing control operations, the CPU is sometimes stopped and thus the dual port RAM is non-operative.

These control operations are **NOT** supported for S155U programmable controllers.

To do control operations one needs to use the CP-511 communication card which can communicate with the CPU even when it is stopped. In this configuration, the host is connected to the CP-535 card via Ethernet. The CP-535 is connected to the CP-511 by an RS-232 cable, and the CP-511 communicates with the CPU via the backplane.

To do control operations, the host sends a series of requests to the CPU by going through the CP-535 and the CP-511 cards. For example, for an upload, the host first finds the type and status of the CPU, then determines what memory records have been programmed in the programmable controller. It then requests the contents of each of these memory records and writes them to a file.

3.2.3.2 Model S135U Communications

Model S135U of the Siemens programmable controller allows for multiple CPU cards within the same programmable controller. To coordinate between the CPUs there is a special Coordinator/MUX card.

Model S135U programmable controllers do not need the CP-511 card. The CPU may be connected directly to the CP-535 using a similar RS-232 cable as in the direct connection above. As such, when the CP-535 is connected directly to the CPU card, all control operations are done only to that CPU as if there are no other CPUs in the programmable controller.

The RS-232 cable may also be connected between the CP-535 card and the Coordinator card. In this configuration, control operations may be done to any of the CPU's in the programmable controller without changing the position of the cable. The Coordinator card is responsible for routing the control operation to the appropriate CPU.

3.2.3.3 Common Steps

Before doing the actual upload, download, start, stop or status, the following steps need to be performed:

- Establish the Transport connection to the special TSAP
- Access the CP-535 card.

  If the programmable controller is an S135U, and if connected with Coordinator, also select CPU
- Find the model, type, and revision of the CPU
- Find the operational status of the CPU

Then, perform the various steps specific to the control operation. When the whole operation is finished, or if there is an error condition, then abort the connection to the special TSAP.

3.2.3.4 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. See the manufacturer's documentation for specific safety precautions.

The following are the steps to start the CPU:

- Finish the common steps outlined above
- Start CPU

The following are the steps to stop the CPU:

- Finish the common steps outlined above
- Stop CPU

Start and Stop Qualifiers

Qualifier	Value	Description
START_OPTIONS	CPUn	For the model S135U, this specifies the CPU to be started
STOP_OPTIONS	CPUn	For the model S135U, this specifies the CPU to be stopped

CLI examples

To start the second CPU of a model 135 programmable controller type the following.

```
DCM> START SIE_135 /START_OPTIONS=CPU2
```

To stop the CPU of a model 150 programmable controller type the following.

```
DCM> STOP SIE_150
```

3.2.3.5 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.

The following are the steps for the upload operation:

- Finish the common steps outlined above
- Find directory layout of first data area
- Read first memory record defined for this area
- Write it to file
- Repeat above two steps for all memory records in this area

- Repeat above four steps for all data areas in the programmable controller

After finishing the common steps, the DAS for Siemens H1 Protocol asks for the directory layout of the memory of the programmable controller. For each data area, it obtains all data records and writes them to file.

The following are the steps for the download operation:

- Finish the common steps outlined above
- Open the OpenVMS file containing the program to be downloaded. Verify that it contains a Siemens program.
- Clear all memory in the CPU
- Read the first record in the file
- If necessary inform the programmable controller that there is a new record in a new data area.
- Repeat above two steps for all records in the file

When downloading to the CPU, one needs to inform the CPU that there is a new record. The CPU makes the proper entry in its directory and readies itself to accept the new record. If for whatever reason it can not do so, it replies with a protocol error and the download operation is terminated.

In this case, the memory contents are left in an unknown state.

When attempting a new download, the memory contents are cleared again, thus retrying the download operation from a cleared condition.

Note that the memory contents of the CPU are cleared first, hence the CPU should not reject the records unless there is an error in the OpenVMS file.

Upload and Download Qualifiers

Qualifier	Value	Description
DEVICE_FILE	CPUn	For the model S135U, this specifies the CPU to be uploaded or downloaded.

CLI examples

To upload the memory contents of the third CPU of a model 135 programmable controller, type the following:

```
DCM> UPLOAD SIE_135 DUA0:[LOGIC_FILES]PROG_3.DAT /dev=CPU3
```

To download to the memory of a CPU of a model 135 programmable controller connected directly to the CP-535 card, type the following:

```
DCM> DOWNLOAD SIE_135 DUA0:[LOGIC_FILES]PROG_ONE.DAT
```

3.2.3.6 Read Status

The read status function issues a diagnostic status request to the device, interprets the device response, and returns the interpretation as a character buffer. An error message is displayed if a device definition does not match the device in the device response. If this occurs, you must correct the device definition before you can perform an upload function or download function for the device.

The following are the steps for the status operation.

First, the common steps listed above are sequentially executed. The common steps above return quite a lot of status information to the user. If the transport connection can be established then the programmable controller is available on the Ethernet, and at least the CP-535 card is functional. If the CPU can be reached, then the CP-511 card or the CPU or the Coordinator are properly connected to the CP-535 and they are all functional. Next, one can determine if the CPU is properly communicating if it returns the type of the CPU and the version level of its software. Following this, one can determine if the CPU is runnning or stopped.

Then, one may request the memory layout of the CPU. This information is useful to quickly glance at the programs and data stored in the programmable controller. This is done by requesting the directory information of all data areas of the programmable controller.

Read Status Qualifiers

Qualifier	Value	Description
QUALIFIER	CPUn	For the model S135U, this specifies the CPU to read status from.
	MEM	Directory layout of memory is returned.
	CPUn,MEM	For the model S135U, this specifies the CPU to return directory information for.

When using the MEM qualifier, one should also specify /FULL to display the entire status.

CLI examples

To display the current status of device SIE_ONE type the following:

```
DCM> READ STATUS SIE_ONE
```

This produces a display similar to the following:

```
 Status of device  SIE_ONE  at 29-DEC-1992 12:15:58.26

 The CP511 board is available
 PLC 135U Type R   PLC Sftwr  v9.0    Comm Sftwr  v9.0
 PLC is stopped
```

To display the current status of the second CPU of a model 135 programmable controller and to display the memory layout of the CPU, type the following:

```
DCM> READ STATUS SIE_ONE /QUAL=(CPU2,MEM) /FULL
```

This produces a display similar to the following:

```
 Status of device  SIE_ONE  at 29-DEC-1992 12:15:58.26

 The CP511 board is available
 PLC 135U Type R   PLC Sftwr  v9.0    Comm Sftwr  v9.0
 PLC is stopped
 PB>     10:0007
 FB>     10:0010     11:0072     12:00AD     13:00C4    120:0107
 FB>    121:012B    122:014F    123:0173    124:018B    125:01A0
 FB>    126:01B5    127:01CD
 OB>      1:01E5     13:01F2     20:0227     21:0238     22:0249
 DB>      0:DD85     10:025A     11:0273     12:0377    100:047B
 DB>    101:0580
```

3.2.4 Read and Write Data

Use the read and write data block functions to read data from and write data to a specific address in device storage. Use read and write data message functions to send and receive messages from the programmable controller. These message functions are available only if the device is "unsolicited".

3.2.4.1 Data Block Access

Use the read data and write data functions to read data from and write data to a specific address in device storage. Address display is specific to each device or device family.

To read data, one has to define the read TSAP. This is done with the /NETADDR definition of the Siemens device. If not already established, the DAS for Siemens H1 Protocol establishes the Transport connection to the READ TSAP of the programmable controller.

Next, the DAS for Siemens H1 Protocol sends a request to the programmable controller to receive the data. When the programmable controller responds, the DAS for Siemens H1 Protocol interprets the response, extracts the data from it, converts it from programmable controller format to host understandable format, and returns the converted data to the requesting application.

From the programmable controller's point of view, the CP-535 card receives the request for data from the user connected to the Ethernet. When the data is available, it is returned to its requestor. The CP-535 card and the CPU communicate via a dual-port RAM which must be properly synchronized between the two. Refer to Siemens documentation for more detailed information.

To write data, one has to define the write TSAP. This is done with the /NETADDR definition of the Siemens device. If not already established, the DAS for Siemens H1 Protocol establishes the transport connection to the write TSAP of the programmable controller.

Next, the DAS for Siemens H1 Protocol converts the data from the application to a format understandable by the programmable controller, and encapsulates it in a request to the programmable controller to write data.

Next, the DAS for Siemens H1 Protocol sends this request to the programmable controller to write the data. When the programmable controller responds, the DAS for Siemens H1 Protocol determines if the data was written successfully.

From the programmable controller's point of view, the CP-535 card receives the request to modify data from the user connected to the Ethernet. The result of the operation (successful or not) is then returned to its requestor. The CP-535 card and the CPU communicate via a dual-port RAM which must be properly synchronized between the two.

Addressing Syntax for Data Block Access

programmable controllers memory locations can be addressed according to the following syntax:

```
<PC_block> : <block_number> : <block_offset>
```

or

```
<PC_block> : <block_number>
```

Table 3–2 shows allowed values for `PC_block`, `block_number` and `block_offset`.

Table 3–2 Data Block Addressing

Block	Block Numbers	Block Offset	Format
DB	1-255	0-2047	STRING[1] BUFFER[2] WORD[3] LONG[4] FLOAT[5]
DX	1-255	0-2047	STRING BUFFER WORD
DE	1-255	0-2047	STRING BUFFER WORD
FB	0-255		STRING BUFFER BYTE
IB	0-127		STRING BUFFER BYTE
OB	0-127		STRING BUFFER BYTE
PB	0-127		STRING BUFFER BYTE
CB	0-255		STRING BUFFER WORD
TB	0-127		STRING BUFFER BYTE
RF	0-127		STRING BUFFER WORD
AS	0-65535		STRING BUFFER WORD
EB	0-255		STRING BUFFER BYTE

[1]No format conversion occurs for string data.

[2]No format conversion occurs for buffer data.

[3]Word data is byte swapped.

[4]Longword data is transferred as two byte swapped words.

[5]Floating point data is converted to/from VAX float format.

Structured data is supported as long as all elements of the structure have the same format. Arrays of a supported format are also allowed.

CLI examples

To read or write to the device type the following:

```
DCM> READ DATA S135/ADDR="DB:100:0"/FORMAT=ARRAY[10]:WORD


DCM> WRITE DATA S150/ADDR="OB:10"/FORMAT:ARRAY[2]:BYTE
```

Addressing Syntax for TI Devices

TI programmable controllers memory locations can be addressed according to the following syntax:

```
<PC_mem><element>
```

Table 3–3 shows allowed values for `PC_mem` and `element`.

Table 3–3 TI Addressing

Memory Type	Element	Format	Description
V	1-10000	WORD	Variable Memory
CR	1-10000	BYTE	Control Registers

CLI examples

To read or write to device TI type the following:

```
DCM> READ DATA TI/ADDR="V1"/FORMAT=ARRAY[10]:WORD


DCM> WRITE DATA TI/ADDR="CR5"/FORMAT:ARRAY[2]:BYTE
```

When performing the read or write to the programmable controller the DAS converts the address described above to an address of the form DB:<block_number>:<block_offset>.

The actual location read in the programmable controller is controlled by the parameters ILAN$H1_TI_V_BASE and ILAN$H1_TI_V_SIZE for variable memory and by the parameters ILAN$H1_TI_CR_BASE and ILAN$H1_TI_CR_SIZE for control registers. The `DB` block number and offset are calculated using the following formulas:

```
block_number = BASE + ((element - 1) / SIZE)

block_offset = element - ((block_number - BASE) * SIZE) - 1
```

See Section 2.3.4 for more information on the TI BASEstar parameters.

3.2.4.2 Read and Write Message

You can also read messages from or send messages to the programmable controller using the BASEstar Classic device connection management read data and write data functions.

The read data and write data message functions allow you to do an acknowledged write, an unacknowledged write or a read operation.

To read or write a message, one has to define the unsolicited TSAP by specifying the name of the unsolicited TSAP in the /NETADDR definition of the Siemens device.

The address data does not match a specific address in device storage. It specifies what kind of operation (read/write acknowledged/unacknowledged) has to be performed.

Note

These operations are only allowed on an unsolicited device.

Acknowledged Write Address Syntax

To perform an acknowledged write, the data address qualifier must have the following format:

```
/ADDRESS=ACK:xxxx
```

xxxx is any positive integer value that fits in a longword. This value is called a "message ID".

CLI example

To write an acknowledged message to the device type the following:

```
DCM> WRITE DATA SIE_ONE/ADDRESS=ACK:587/FORMAT=STRING
```

The DAS for Siemens H1 Protocol sends a buffer to the programmable controller to write the data.

This buffer consists of a message header followed by the data to be written. The message header has the following format:

- First longword : the message ID as given in the /ADDRESS qualifier.
- Second longword: the request ID, internally allocated by the DAS.

The data part of this buffer has been converted to a format understandable by the programmable controller as specified in the format of the request.

The programmable controller must be programmed to understand the message header and to acknowledge the data written by sending back the message header.

The message ID can be used by the programmable controller to find out what kind of data has been written.

When the programmable controller responds, the DAS for Siemens H1 Protocol checks if the message ID and the request ID match a pending write request. If a matching write request has been found, the DAS will consider it as acknowledged and will return the acknowledgment to the application.

See Appendix C for a complete description of data exchange over the unsolicited TSAP.

Unacknowledged Write Address Syntax

To perform an unacknowledged write, the data address qualifier must have the following format:

```
/ADDRESS=NAK:xxxx
```

Where xxxx is any positive integer value that fits in a longword. This value is called a "message ID".

CLI example

To write an unacknowledged message to the device type the following:

```
DCM> WRITE DATA SIE_ONE/ADDRESS=NAK:15349/FORMAT=ARRAY[10]:WORD
```

The DAS for Siemens H1 Protocol will send a buffer to the programmable controller to write the data.

This buffer consists of a message header followed by the data to be written to the programmable controller. The message header has the following format:

- First longword : the message ID as given in the /ADDRESS qualifier.
- Second longword: 0

The data part of this buffer has been converted to a format understandable by the programmable controller as specified in the format of the request.

In this case, when the second longword of the message header has been set to 0, the programmable controller must understand that it has no acknowledgement to return to the DAS for Siemens H1 Protocol.

Note

Do not attempt to read using an unacknowledge write address. Doing so will result in an error being returned to the user.

See Appendix C for a complete description of data exchange over the unsolicited TSAP.

Acknowledged Read Address Syntax

A message read operation is always acknowledged. To perform a read operation, the data address qualifier must have the following format:

```
/ADDRESS=ACK:xxxx
```

xxxx is any positive integer value that fits in a longword. This value is called "message ID".

CLI example

To read a message from the device type the following:

```
DCM> READ DATA SIE_ONE/ADDRESS=ACK:1/FORMAT=STRING
```

The DAS for Siemens H1 Protocol will send a request to the programmable controller to receive the data.

This request only consists of a message header having the following format:

- First longword : the message ID as given in the /ADDRESS qualifier.
- Second longword: the request ID, internally allocated by the DAS.

The programmable controller must be programmed to understand the message header. In this case, when the message received by the programmable controller is only 2 longwords long, it has to answer by returning to the DAS the message header, followed by the requested data.

The message ID may be used by the programmable controller to find out what kind of data has been requested.

When the programmable controller responds, the DAS for Siemens H1 Protocol will check if the message ID and the request ID match a pending read request; it will extract the data from the response, convert it from programmable controller format to host understandable format as specified by the format of the request and return the converted data to the requesting application.

See Appendix C for a complete description of data exchange over the unsolicited TSAP.

3.2.5 Send and Receive Message

To send or receive a message, one has to define the unsolicited TSAP by specifying the name of the unsolicited TSAP in the /NETADDR definition of the Siemens device.

Note

These operations are only allowed on a device that is NOT defined as unsolicited.

To send data, the application program calls routine ILAN$DEVICE_SPECIFIC specifying function code 51 and a buffer containing the data packet to be sent.

The DAS for Siemens H1 Protocol then writes this buffer on the unsolicited channel.

To receive data, the application program calls routine ILAN$DEVICE_SPECIFIC specifying function code 50 and a buffer to receive the data packet.

The DAS for Siemens H1 Protocol then tries to read data from the unsolicited channel.

When the programmable controller sends its data packet, the read operation completes and the data is returned to the application program.

See Appendix B for program examples. These programs are also available at SYS$COMMON:[SYSHLP.EXAMPLES.DCM_H1].

ILAN$DEVICE_SPECIFIC function codes are defined in the include file SYS$LIBRARY:DCM_H1$DEF.H for "C" program users.

3.3 Automatic Data Collection

3.3.1 Unsolicited Data

Unsolicited data involves a device sending data to a physical point in BASEstar Classic device connection management BASEstar Classic device connection management requesting it. BASEstar Classic device connection management can accept unsolicited values from any device, as long as the device is defined as unsolicited and physical points are defined with unsolicited information.

All unsolicited data is sent over the unsolicited TSAP which is defined by specifying the name of the unsolicited TSAP in the /NETADDR definition of the Siemens device.

Siemens devices are capable of sending unsolicited messages to the host. The DAS for Siemens H1 Protocol allows you to define BASEstar Classic device connection management physical points for collecting unsolicited data. If the device sends an unsolicited message, the DAS forwards the message to BASEstar Classic device connection management.

An unsolicited message consists of a message header followed by the unsolicited data. The message header must have the following format:

- First longword : any positive integer value
- Second longword : 0

To define a physical point having an unsolicited ID, use the following qualifier: `/UNSOLICITED=xxxx`

Where xxxx is any positive integer value that fits in a longword and that will uniquely identify an unsolicited message.

The /ADDRESS qualifier of an unsolicited physical point can contain any valid point address (See Section 3.2.4).

Unsolicited messages are received in the DAS by posting an unsolicited read on the unsolicited TSAP. When the device sends an unsolicited message over this TSAP, the read is completed and sent to BASEstar Classic device connection management.

If the first longword of the received message matches a physical point unsolicited ID and if the second longword of the received message has been set to 0, then the BASEstar Classic device connection management will be updated with the data part of the message.

3.3.2 Pollsets

BASEstar Classic device connection management physical points may be grouped together into pollsets to optimize data collection.

Data can be polled on the read TSAP or on the unsolicited TSAP.

Pollsets created for the Siemens 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 BYTE cannot be members of the same pollset. Physical points of the same data type with varying element counts can reside in the same pollset.
- A pollset used with the unsolicited TSAP can only contain one physical point.
- Physical points with different block types cannot be part of the same pollset.
- Physical points with different block numbers cannot be part of the same pollset.

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

A
Logged Messages

The messages in the following sections are logged to the BASEstar Classic history file by the DAS for Siemens H1 Protocol. 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 Siemens H1 Protocol use event class 21 and event type 30.

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

```
$ BSTAR
BSTAR> SHOW HISTORY/EVENT=21.30.*
```

A.1 NI Logged Messages

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

Matching unsolicited request for message ID !UL was not found.

Explanation: Error. The DAS received a message on the unsolicited TSAP, but the second longword in the message did not contain the ID of a valid request block. The message logs the ID of the message found in the first longword of the message. The most likely cause for this error is that the read on the unsolicited TSAP timed out before the response message was received from the programmable controller.

User Action: If the message timed out, then increase the timeout on the path definition for the device. If the wrong message ID was sent, then modify the program in the programmable controller to send the correct message ID.

Unsolicited request for message ID !UL timed out.

Explanation: Error. The read on the unsolicited TSAP for the indicated message timed out.

User Action: Increase the timeout on the path definition for the device or modify the programmable controller to respond to the message in a timely manner.

Point with message ID !UL not added to pollset. Only one point allowed on an unsolicited TSAP.

Explanation: Error. An attempt was made to add more than one physical point to a pollset where the physical point address indicates that the data is being read from the unsolicited TSAP. Since the data for this physical point type is a message and not a programmable controller memory location, the DAS cannot read more than one physical point at a time.

User Action: Create a separate pollset for the physical point.

Invalid parameter ILAN$H1_TI_V_BASE : must be DB:xxx.

Explanation: Error. The value for the BASEstar parameter ILAN$H1_TI_V_BASE is incorrect. Either the parameter does not start with "DB:" or the base value is not an integer number.

User Action: Modify the parameter to use the proper form.

Parameter ILAN$H1_TI_V_BASE = !AD is out of limits.

Explanation: Error. The value of the parameter ILAN$H1_TI_V_BASE is larger than that allowed. The allowed range is 0-2047.

User Action: Modify the parameter to be within the allowed range.

Invalid parameter ILAN$H1_TI_CR_BASE : must be DB:xxx.

Explanation: Error. The value for the BASEstar parameter ILAN$H1_TI_CR_BASE is incorrect. Either the parameter does not start with "DB:" or the base value is not an integer number.

User Action: Modify the parameter to use the proper form.

Parameter ILAN$H1_TI_CR_BASE = !AD is out of limits.

Explanation: Error. The value of the parameter ILAN$H1_TI_CR_BASE is larger than that allowed. The allowed range is 0-2047.

User Action: Modify the parameter to be within the allowed range.

Block offset is out of limits for address !AD.

Explanation: Error. The physical point address block offset value is outside the allowed limits. See Table 3–2 for valid block offset values.

User Action: MOdify the physical point address to be within valid limits.

Address !AD does not contain ':'.

Explanation: Error. The indicated physical point address syntax is incorrect.

User Action: Use the correct syntax when creating the physical point address. See Addressing Syntax for Data Block Access for information on the proper block syntax.

PC block is invalid for address !AD.

Explanation: Error. In the indicated physical point address, the programmable controller block type is not a valid block type. See Table 3–2 for valid block types.

User Action: Modify the address to use a valid block type.

Message ID is not numeric for address !AD.

Explanation: Error. In the indicated physical point address, the message ID contains non-numeric characters.

User Action: Modify the address to use all numeric characters in the message ID portion of the address.

Message ID must not be zero for address !AD.

Explanation: Error. In the indicated physical point address, the message ID is zero.

User Action: Modify the address to use a non-zero value in the message ID portion of the address.

Block number is not numeric for address !AD.

Explanation: Error. In the indicated physical point address, the block number contains non-numeric characters (or is not a number). See Addressing Syntax for Data Block Access for information on the proper block syntax.

User Action: Modify the address to use a number for the block number.

A block offset was supplied, but none is required for address !AD.

Explanation: Error. In the indicated physical point address, a block offset value was given, but for this block type, no block offset is expected. See Table 3–2 for valid block offset values.

User Action: Modify the address to not specify a block offset or use a block type that expects a block offset.

No block offset was supplied, but one is required for address !AD.

Explanation: Error. In the indicated physical point address, no block offset value was given, but for this block type, a block offset is expected. See Table 3–2 for valid block offset values.

User Action: Modify the address to specify a block offset or use a block type that doesn't expect a block offset.

Block number is out of limits for address !AD.

Explanation: Error. In the indicated physical point address, the block number is not valid for the block type given. See Table 3–2 for valid block offset values.

User Action: Modify the address to specify a valid block number.

Block offset is not numeric for address !AD.

Explanation: Error. In the indicated physical point address, the block offset contains non-numeric characters (or is not a number). See Addressing Syntax for Data Block Access for information on the proper block syntax.

User Action: Modify the address to use a number for the block offset.

All points in a pollset must have the same format.

Explanation: Error. An attempt was made to add a physical point to a pollset and the format of the new physical point does not match the format of the existing physical point(s) in the pollset.

User Action: Modify the format of the physical point being added to match the format of the physical point(s) already in the pollset.

All points in a pollset must have the same block type.

Explanation: Error. An attempt was made to add a physical point to a pollset and the block type of the new physical point does not match the block type of the existing physical point(s) in the pollset.

User Action: Modify the block type of the physical point being added to match the block type of the physical point(s) already in the pollset.

All points in a pollset must have the same block number.

Explanation: Error. An attempt was made to add a physical point to a pollset and the block number of the new physical point does not match the block number of the existing physical point(s) in the pollset.

User Action: Modify the block number of the physical point being added to match the block number of the physical point(s) already in the pollset.

Not all fields in the structure have the same format for address !AD.

Explanation: Error. In the physical point with the indicated address, the format is of type STRUCTURE, but not all fields in the structure have the same format. The DAS does not support mixing of fields of different data types within a structure.

User Action: Modify the format of the physical point to use fields of all the same format.

Structure not aligned on an even structure boundary in pollset.

Explanation: Error. An attempt was made to add a physical point with a structure format to a pollset and the new structure is not aligned on an even structure boundary.

User Action: Modify the address of the physical point so that it fits on an even structure boundary relative to the other physical point(s) in the pollset.

Longword or floating point not aligned on an even boundary in pollset.

Explanation: Error. An attempt was made to add a physical point with a longword or floating point format to a pollset and the physical point is not aligned on an even four byte boundary.

User Action: Modify the address of the physical point so that it fits on an even four byte boundary relative to the other physical point(s) in the pollset.

dcm_h1$k_receive_message not allowed on an unsolicited device.

Explanation: Error. An attempt was made to use the ILAN$DEVICE_ SPECIFIC function with a device that is defined as "unsolicited". ILAN$DEVICE_SPECIFIC functions are not allowed on unsolicited devices.

User Action: Modify the device to be nounsolicited.

dcm_h1$k_send_message not allowed on an unsolicited device.

Explanation: Error. An attempt was made to use the ILAN$DEVICE_ SPECIFIC function with a device that is defined as "unsolicited". ILAN$DEVICE_SPECIFIC functions are not allowed on unsolicited devices.

User Action: Modify the device to be nounsolicited.

Device must be configured as unsolicited to perform this function.

Explanation: Error. An attempt was made to perform an acknowledged or unacknowledged write or an acknowledged read and the device is not configured as "unsolicited".

User Action: Modify the device to be unsolicited.

Expected message ID !UL but message ID !UL was received for unsolicited read.

Explanation: Error. The message ID in the message that was received in response to an acknowledged write or an acknowldeged read does not contain the expected message ID. The message ID in the message sent should match the message ID in the message received. Either the message ID is incorrect or the request ID that was sent in the message was not copied properly to the response message.

User Action: Modify the program in the programmable controller to properly send the correct request and message IDs in response to an acknowledged write or read request.

An unacknowledged read is not supported.

Explanation: Error. An attempt was made to read from a physical point that has an address of NAK:XXX.

User Action: Do not read from a physical point with a NAK:XXX address. If reading from a logical point connected to such a physical point, use the /VALUE=LAST qualifier to prevent an attempt to read from the device.

Format defined for address !AD allowed in DB area only.

Explanation: Error. In the physical point with the indicated address, the format is either a longword or floating point format and the address is not a DB block address. Longwords and floating point values can only be read from the DB block area.

User Action: Modify the address of the physical point to be a "DB:XX:XX" type address.

TI CR registers must use a byte format for address !AD.

Explanation: Error. Control register formats must be byte or unsigned byte. No other format type is allowed.

User Action: Modify the physical point to use a signed or unsigned byte format.

TI V regiters must use a word format for address !AD.

Explanation: Error. Variable reigster formats must be word or unsigned word. No other format type is allowed.

User Action: Modify the physical point to use a signed or unsigned word format.

IO_SIZE too small to fit the requested data.

Explanation: Error. The size of the data (plus overhead) being read or written is larger than will fit into the space allocated in the DAS. The DAS allocates space based on the IO_SIZE parameter on the path definition for the device.

User Action: Modify the path definition I/O size to be a value that is large enough for the data plus overhead.

Error code !XB (hex) received from the programmable controller.

Explanation: Error. A response was received from the programmable controller, but an error code was received encoded in the message.

User Action: Consult Siemens documentation to determine the cause of the error and to take corrective action.

Source/destination type illegal.

Explanation: Error. A response was received from the programmable controller, but an error code of ’01’ was received in the message. This error indicates that the source/destination address is not legal for this programmable controller.

User Action: Delete the physical point and recreate it using an address that is valid for this programmable controller.

DB/DX not present or illegal.

Explanation: Error. A response was received from the programmable controller, but an error code of ’02’ was received in the message. This error indicates that the requested DB/DX block requested is not present in the programmable controller.

User Action: Delete the physical point and recreate it using an address that is valid for this programmable controller.

DB/DX too short (start address + length > area).

Explanation: Error. A response was received from the programmable controller, but an error code of ’03’ was received in the message. This error indicates that the starting address is present in the programmable controller, but the length of the request extends into non-existent memory.

User Action: Delete the physical point and recreate it using an address that is valid for this programmable controller.

Access to area not possible for user.

Explanation: Error. A response was received from the programmable controller, but an error code of ’04’ was received in the message. This error indicates that the address is present in the progammable controller, but that the user is being denied access.

User Action: Delete the physical point and recreate it using an address that is valid for this programmable controller.

No matching unsolicited point ID found for message ID !UL.

Explanation: Error. An unsolicited message was received from the programmable controller, but no matching unsolicited ID was found that matched the message ID in the message. It is possible for this message to occur if the programmable controller starts sending immediately after connection and the DAS doesn’t have enough time to set up the unsolicited points. See Section 3.3.1 for more information on unsolicited data collection.

User Action: Create a physical point with a matching unsolidited ID or modify the programmable controller logic to send a message with a message ID of an existing physical point.

CP-511 board is not connected.

Explanation: Error. An attempt was made to perform an operation on the control TSAP (upload, download, etc.) and the CP-511 card was not connected to the CPU. See Section 3.2.3.1 for more details.

User Action: Attach the CP-511 card to the CPU using the proper cable before performing the operation.

No room in temporary buffer for reading block data.

Explanation: Error. When doing an UPLOAD or READ STATUS /QUAL=MEM request, the programmable controller is returning more block information than will fit in the buffer allocated for the data (1024 bytes).

User Action: Reduce the number of blocks in the programmable controller.

File !AD does not match the CPU type.

Explanation: Error. The CPU model of the programmable controller this file was uploaded from does not match CPU model of the programmable controller being downloaded to.

User Action: Download a file compatible with this model of programmable controller.

Device returned X'!XB' after sending block advise request.

Explanation: Error. When downloading blocks to a programmable controller, the controller returned the indicated response rather than the expected response indicating the programmable controller was ready to accept the next block.

User Action: Retry the operation.

Fatal error on the command TSAP.

Explanation: Error. When performing a request on the command TSAP (upload, download, status), the programmable controller returned a response that indicated it was unable to complete the requested operation.

User Action: Retry the operation. If the operation does not succeed, then consult Siemens documentation to determine the cause of the failure.

A.2 PE Logged Messages

The following messages are logged by the Protocol Emulator (PE).

Address !AD must have a '%' seperating the NAP and ethernet address.

Explanation: Error. The network address for the device is incorrect. See Device address format for information on the proper syntax for the network address.

User Action: Modify the network address using the correct syntax.

Address !AD does not contain a valid ethernet address.

Explanation: Error. The ethernet address part of the network address for the device does not contain twelve hexadecimal digits. See Device address format for information on the proper syntax for the network address.

User Action: Construct a network address that contains an ethernet address of twelve hexadecimal digits.

TSAP list must start with a '(' and end with a ')'.

Explanation: Error. The list of TSAPs in the network address for the device is not delineated by parenthesis. See Device address format for information on the proper syntax for the network address.

User Action: Construct a network address that contains a list of TSAPs delineated by parenthesis.

Error deassigning a channel for TSAP !AD.

Explanation: Error. An error occurred when deasigning an OpenVMS device channel assigned to the indicated TSAP.

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

Error translating logical name !AD.

Explanation: Error. The DAS expected (based on the network address syntax) to retrieve some or all of the TSAPs from the indicated logical name, but received an error when attempting to do so.

User Action: Verify that the logical name exists and is in a scope that can be resolved by the ILAN$DEVSRV process.

Syntax for TSAP list !AD is not valid.

Explanation: Error. The indicated list of TSAPs is not formatted correctly. Either expected tokens are missing or a TSAP other than R, W, U, C or O has been specified. See Device address format for more information on the proper syntax for the network address.

User Action: Use the proper syntax for the TSAP list when entering the network address.

Error assigning a channel for TSAP !AD.

Explanation: Error. An error occurred when calling SYS$ASSIGN for the OSI pseudo device.

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

Error getting timeout value from the path.

Explanation: Error. An error occurred when calling ILAN$$SPT_GET_ PORT_INFO to retrieve the timeout value from the path definition.

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

Error in accessing TSAP !AD.

Explanation: Error. An error occurred when calling SYS$QIO IO$_ACCESS for the indicated TSAP. This call attempts to establish a connection to the indicated TSAP.

User Action: Examine the error returned from the call and take action based on the error found. Make sure that TSAPs are properly configured on the device definition and on the programmable controller. Make sure that the programmable controller communication card is properly configured to accept the connection from the indicated TSAP.

QIO error from !AD request.

Explanation: Error. A QIO error when performing the indicated function.

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

OSIT error on TSAP !AD.

Explanation: Error. A QIO error occurred when performing an operation on the indicated TSAP. The OSI Transport specific error is logged in this message. See *DECnet Plus for OpenVMS Programming* Appendix A for a list of OSI Transport specific error codes and their meanings.

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

Error initiating deaccess request, TSAP = !AD.

Explanation: Error. An error occurred when calling SYS$QIO IO$_ DEACCESS | IO$M_ABORT for the indicated TSAP.

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

!AD request, TSAP is not configured.

Explanation: Error. For the indicated request, the corresponding TSAP has not been configured in the device's network address. (For example, a read request requires that the read TSAP be configured.)

User Action: Configure the required TSAP in the network address and in the corresponding programmable controller.

!AD request, TSAP has not been accessed.

Explanation: Error. The TSAP is configured, but has not been accessed (connected). The connection may have disconnected, so the TSAP is no longer available.

User Action: If the connection disconnected, then the DAS will automatically try to reconnect. If the connection fails, then take action based on the connection failure reason.

Unsolicited TSAP !AD set to permanent.

Explanation: Informational. The unsolicited TSAP was configured as a temporary TSAP. The DAS does not allow the unsolicited TSAP to be temporary.

User Action: None.

Disconnect received.

Explanation: Informational. The DAS received a disconnect on one or more of the currently connected TSAPs. The DAS will abort the connection on all TSAPs.

User Action: None.

Network address !AD, connection completed.

Explanation: Informational. All permanently configured TSAPs for the indicated network address have successfully connected.

User Action: None.

Network address !AD, disconnection completed.

Explanation: Informational. All permanently configured TSAPs for the indicated network address have successfully disconnected.

User Action: None.

B

Sending and Receiving Data Using ILAN$DEVICE_SPECIFIC

This service is used to send or receive data packets to or from Siemens programmable controllers.

To send or receive a message, one has to define the unsolicited TSAP by specifying the name of the unsolicited TSAP in the /NETADDR definition of the Siemens device.

Note

You cannot use the ILAN$DEVICE_SPECIFIC routine to send and receive data on a device defined as unsolicited.

You can use the following program to send data to the programmable controller, using an unsolicited TSAP.

```
#include stdio
#include stdlib
#include ssdef

#include <ilan_defs>
#include <ilan_services>
#include <dcm_h1$def>

main ()
    {
    long int ret;
    char *device_name = "S150_DEVICE";
    long int send_function = dcm_h1$k_send_message; /* Send message */
    short int ret_length;
    unsigned char request_buffer[512];
    unsigned char *null_buffer = NULL;
    short int request_buffer_len;
    short int response_buffer_len;

    request_buffer_len = 10;
    response_buffer_len = 0;
    ret = ilan_device_specific (0,
    device_name,
    &send_function,
    &request_buffer_len,
    (struct itmlst *) request_buffer,
    &response_buffer_len,
    (struct itmlst *) null_buffer,
    &ret_length);
    exit (ret);
    } /* End main */
```

Sending and Receiving Data Using ILAN$DEVICE_SPECIFIC

You can use the following program to receive data from the programmable controller, using an unsolicited TSAP.

```
#include stdio
#include stdlib

#include <ilan_defs>
#include <ilan_services>
#include <dcm_h1$def>
main ()
    {
    long int ret;
    char *device_name = "S150_DEVICE";
    long int rcv_function = dcm_h1$k_receive_message; /* Receive message */
    short int ret_length;
    unsigned char request_buffer[512];
    unsigned char response_buffer[512];
    unsigned char *null_buffer = NULL;
    short int request_buffer_len;
    short int response_buffer_len;

    request_buffer_len = 0;
    response_buffer_len = sizeof (response_buffer);
    ret = ilan_device_specific (0,
                                device_name,
                                &rcv_function,
                                &request_buffer_len,
                                (struct itmlst *) null_buffer,
                                &response_buffer_len,
                                (struct itmlst *) response_buffer,
                                &ret_length);
    exit (ret);
    } /* End main */
```

C

Data exchange using the unsolicited TSAP

The method of data exchange using the unsolicited TSAP is different depending on whether the device has been defined as unsolicited or not.

If the device is **not** an unsolicited device, one can send or receive data to or from Siemens programmable controllers using the unsolicited TSAP with the ILAN$DEVICE_SPECIFIC routine. Refer to Appendix B for more information.

If the device **is** defined as unsolicited (by the /UNSOLICITED qualifier in the CREATE DEVICE command), the unsolicited TSAP can be used:

- by the DAS to perform a read operation, an acknowledged write operation or an unacknowledged write operation.
- by the programmable controller to send unsolicited data to the DAS for Siemens H1 Protocol.

All these operations are described in this appendix.

The unsolicited TSAP is defined by specifying the name of the unsolicited TSAP in the /NETADDR definition of the Siemens device.

Note

For all the following operations, the device must be defined as unsolicited.

When the DAS for Siemens H1 Protocol uses the unsolicited TSAP to exchange data with a device defined as **unsolicited**, it sends for each request a message header, made of 2 longwords, that may be followed by some specific data.

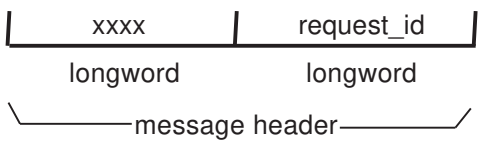
The programmable controller must be programmed to understand this message header, and to cooperate with the DAS for Siemens H1 Protocol.

C.1 Read Operation

For example:

```
DCM> READ DATA S135/ADDR=ACK:xxxx/FORMAT=ARRAY[10]:WORD
```

The Siemens sends the following buffer to the programmable controller over the unsolicited TSAP:



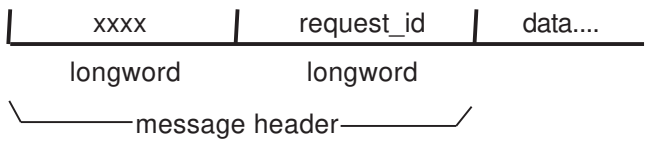
Where:

- xxxx is any positive integer value that fits in a longword.

  This value is given by the /ADDR qualifier. It is called the message ID.

- request_id is the identifier for the read request block, as it has been allocated by the DAS for Siemens H1 Protocol.

The message ID "xxxx" may be used by the program in the programmable controller to find out what kind of data is expected by the DAS for Siemens H1 Protocol. The programmable controller must answer by returning the message header sent by the DAS, followed by the requested data:

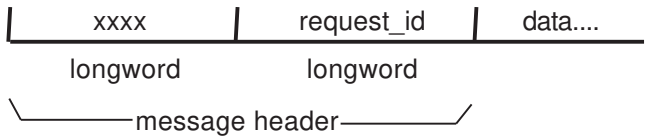


C.2 Acknowledged Write Operation

For example:

```
DCM> WRITE DATA S135/ADDR=ACK:xxxx/FORMAT=ARRAY[3]:WORD
```

The Siemens sends the following buffer to the programmable controller over the unsolicited TSAP:



Where:

- xxxx is any positive integer value that fits in a longword.

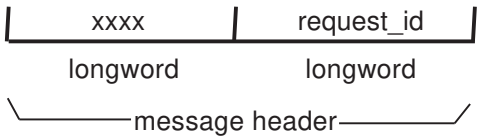
  This value is given by the /ADDR qualifier. It is called the message ID.

- request_id is the identifier for the write request block, as it has been allocated by the DAS for Siemens H1 Protocol.

- data represents the data sent to the programmable controller.

  Inin our example, it would be a 3 words long buffer.

The message ID "xxxx" may be used by the program in the programmable controller to find out what kind of data has been sent by the DAS for Siemens H1 Protocol. The programmable controller must acknowledge the write request by returning the message header:

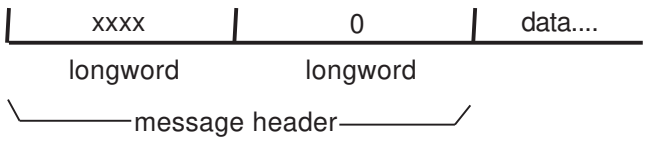


C.3 Non Acknowledged Write Operation

For example:

```
DCM> WRITE DATA S135/ADDR=NAK:xxxx/FORMAT=ARRAY[5]:LONGWORD
```

The Siemens sends the following buffer to the programmable controller over the unsolicited TSAP:



Where:

- xxxx is any positive integer value that fits in a longword.

  This value is given by the /ADDR qualifier. It is called the message ID.

- data represents the data sent to the programmable controller

  In our example, it would be a 5 longwords long buffer.

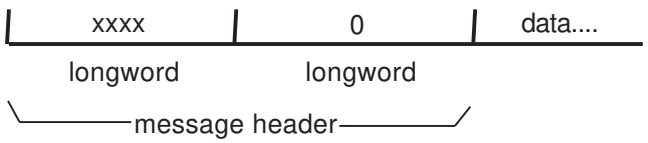
When the second longword sent by the DAS for Siemens H1 Protocol is set to 0, the programmable controller must understand that this involves a non acknowledged write operation, and that it has not to return any answer.

The message ID "xxxx" may be used by the program in the programmable controller to find out what kind of data has been sent by the DAS for Siemens H1 Protocol.

C.4 Unsolicited messages

Unsolicited data involves a device sending a physical point to BASEstar Classic device connection management without BASEstar Classic device connection management requesting it. BASEstar Classic device connection management accept unsolicited values from any device, as long as the device is defined as unsolicted and the physical point is defined with unsolicited information.

When the device sends an unsolicited message, it must use the following format:



Where:

- xxxx is any positive integer value that fits in a longword.
- the second longword of the message header must be set to 0.
- data represents the unsolicited data sent by the programmable controller.

When the DAS for Siemens H1 Protocol receives a message on the unsolicited TSAP, it first checks the second longword of this message: if it is zero, the message is unsolicited.

Data exchange using the unsolicited TSAP
C.4 Unsolicited messages

The DAS for Siemens H1 Protocol then checks if a physical point has been defined with an unsolicited ID equal to "xxxx", first longword of the unsolicited message header.

If this is the case, the data part of the unsolicited message is copied into the physical point.

Such a physical point could have been created with following command:

```
DCM> CREATE PHYP U1/DEV=S135/UNSOLICITED="xxxx" ...
```

Index

A

Accessing DAS functions, 3–1
Addressing syntax, 3–8
  TI, 3–10
Automatic data collection, 3–13
  pollsets, 3–14
  unsolicited, 3–13

C

Communication card
  CP-511, 3–2
  CP-535, 3–2
Configuration file
  device records, 2–8
  path records, 2–7
  type records, 2–7
Configuration File
  editing, 2–7
Configuring parameters
  ILAN$H1_TI_CR_BASE, 2–13
  ILAN$H1_TI_CR_SIZE, 2–13
  ILAN$H1_TI_V_BASE, 2–13
  ILAN$H1_TI_V_SIZE, 2–13
  ILAN$MAX_SPT_REQUESTS, 2–12
Control Operations, 3–4
  Common Steps, 3–4
  Communications, 3–4
  Model S135U, 3–4
CP-511
  communication card, 3–2
CP-535
  communication card, 3–2

D

DAS
  See Device access software
DECnet Plus parameters, 2–10
DECnet Plus Parameters:
  DECnet Plus, 2–10
Device Access Software
  accessing functions, 3–1
  description of, 1–1
  devices, 1–3
  functions, 1–2
Device Access Software (cont'd)
  installation requirements, 2–1
  installion of, 2–1
  overview, 1–1
  supported devices, 1–2, 3–1
  supported functions, 1–2, 3–1
  using, 3–1
Device address format, 2–8
Device records, 2–8
Devices
  communications, 1–1
Disk space required, 2–2
Download function, 3–5

F

Files
  created during installation, 2–5
Functions, 1–3, 3–1
  download, 3–5
  read data, 3–8
  read data block, 3–8
  read message, 3–10
  read status, 3–6
  start, 3–5
  stop, 3–5
  upload, 3–5
  write data, 3–8
  write data block, 3–8
  write message, 3–10

H

Hardware
  required, 2–1

I

Installation, 2–1
  files created, 2–5
  messages, 2–5
  post installation tasks, 2–6
  procedure, 2–3
  requirements, 2–1

M

Messages
  installation, 2–5
  logged, A–1

O

OSI Tracing, 2–14

P

Path records, 2–7
Plant floor equipment
  setting up, 2–14
Pollsets, 3–14
Post installation tasks, 2–6
Problem reporting, 2–14
Product failure, 2–14

R

Read data block
  addressing syntax, 3–8
  TI addressing syntax, 3–10
Read data function, 3–8, 3–10
Read status function, 3–6

S

Software
  required, 2–2
SPT block usage, 2–12
Start function, 3–5
Stop function, 3–5

T

TI
  addressing syntax, 3–10
  address translation, 3–10
  BASEstar parameters, 2–13
Tracing
  OSI messages, 2–14
TSAP
  configuring, 2–8
  permanent, 2–9
  temporary, 2–9
Type records, 2–7

U

Unsolicited data, 3–13
Unsolicited TSAP
  receive message, 3–13
  send message, 3–13
Upload function, 3–5

W

Write data block
  addressing syntax, 3–8
  TI addressing syntax, 3–10
Write data function, 3–8, 3–10