= PMS 400
= PMS173.

Networks · Communications

# DSV11
# Synchronous Device
# Driver Manual

digital™
software

# DSV11 Synchronous Device Driver Manual

Order No. AA-JG79B-TE

| | |
|---|---|
| Revision/Update Information: | This is a new manual |
| Operating System and Version: | VAX/VMS V4.7 |
| Software Version: | DSV11 Synchronous Device Driver V1.0 |

digital™

First Printing, April 1988



The READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.



This manual was produced by the Wide Area Communications Environment group in Reading, England.

# Contents

## Figures



## Tables

# How to Use This Manual

## Manual Objectives

This manual describes how to install the DSV11 synchronous device driver (referred to in the manual as the DSV11 driver) on a VAX/VMS system.

The manual also explains how to control the DSV11 driver through the VAX/VMS operating system using the SQIO system services. This manual does not provide information on all aspects of VAX/VMS input/output (I/O) operations.

## Intended Audience

The manual is intended for:

- Anyone installing the DSV11 driver on a VAX/VMS system.
- System programmers who wish to use the DSV11 driver directly.

System programmers are expected to have some experience with an assembly language, such as VAX MACRO, to understand the examples in this book.

## Structure of this Document

There are five chapters and four appendixes: -

- Chapters 1 through 5 describe how to install and use the DSV11 driver:
  - — Chapter 1 introduces the DSV11 device and the DSV11 driver.
  - — Chapter 2 describes how to install the DSV11 driver.
  - — Chapter 3 explains how to use the DSV11 driver I/O function codes with the SQIO system services.
  - — Chapter 4 shows how to obtain information about the DSV11 device, the DSV11 driver characteristics, and error returns using the $GETDVI system service.
  - — Chapter 5 describes the DSV11 driver I/O status block (IOSB).
- Appendixes A through D contain reference information:
  - — Appendix A lists the DSV11 driver I/O function codes.
  - — Appendix B shows the modem control state transitions.
  - — Appendix C describes how to tune your system to avoid heavy CPU impact at high packet rates and to avoid unnecessary timeouts when running full-duplex DDCMP at very high or very low speeds.
  - — Appendix D contains an example DSV11 driver program using the I/O function codes described in the manual.

## Associated Documents

For reference information, see the following documents:

- *VAX/VMS General User Volume* — contains a complete list of all VAX/VMS documents and a master index of all topics discussed in the VAX/VMS document set.
- *VAX/VMS Networking Manual.*
- *VAX/VMS System Services Reference Manual.*
- *Guide to Programming on VAX/VMS.*
- *Guide to VAX/VMS Software Installation.*
- *Guide to VAX/VMS System Management and Daily Operations.*
- *VAX/VMS System Messages and Recovery Procedures Reference Manual.*

## Conventions Used in this Document

| Convention | Meaning |
|---|---|
| [ ] | Brackets in QIO requests enclose optional arguments. For example:<br>`IO$_SETCHAR P1,[P2],P3,[P4]` |
| ... | Horizontal ellipses indicate that irrelevant characters or QIO arguments have been omitted. For example:<br>This file defines many (but not all) of the XFS... symbolic names described in this section.<br>Vertical ellipses in coding examples indicate that irrelevant lines of code have been omitted. For example:<br>`LOGNAM: .ASCID  SYS$INPUT`<br>`.`<br>`.`<br>`.`<br>`;DETERMINE TERMINAL NAME`<br>`$GETDVI_S -`<br>`DEVNAM=LOGNAM, -`<br>`ITMLST=DVI`<br>Hyphens in coding examples indicate that additional arguments to the QIO request are provided on the following line(s). For example:<br>`CMDFAB:  $FAB  fac=put,fnm=sys$output:, -`<br>`mrs=132,rat=cr,fsz=var` |
| Red print | Indicates text that you enter. |
| `$ DIR` | Dot matrix indicates text that appears on the screen. |
| *time* | Italics indicate variable information. |
| <RET> | Indicates that you should press the RETURN key. |
| <CTRL/Z> | Indicates that you should simultaneously press the CTRL key and the keyboard character shown (in this case Z). |
| numbers | Unless otherwise noted, all numbers in the text are decimal. Nondecimal radixes — binary, octal, or hexadecimal — are explicitly indicated in the coding examples. |

# 1

# Introduction to the DSV11

## 1.1 DSV11 Functions and Capabilities

The DSV11 device is a Direct Memory Access (DMA) communications adapter for MicroVAX II (Q22-bus) processors. The DSV11 driver (SJDRIVER) transmits and receives framed messages to provide an interface between the MicroVAX processor and other devices compatible with these protocols:

- DDCMP
- HDLC (LAPB and LAPBE)
- SDLC
- BISYNC

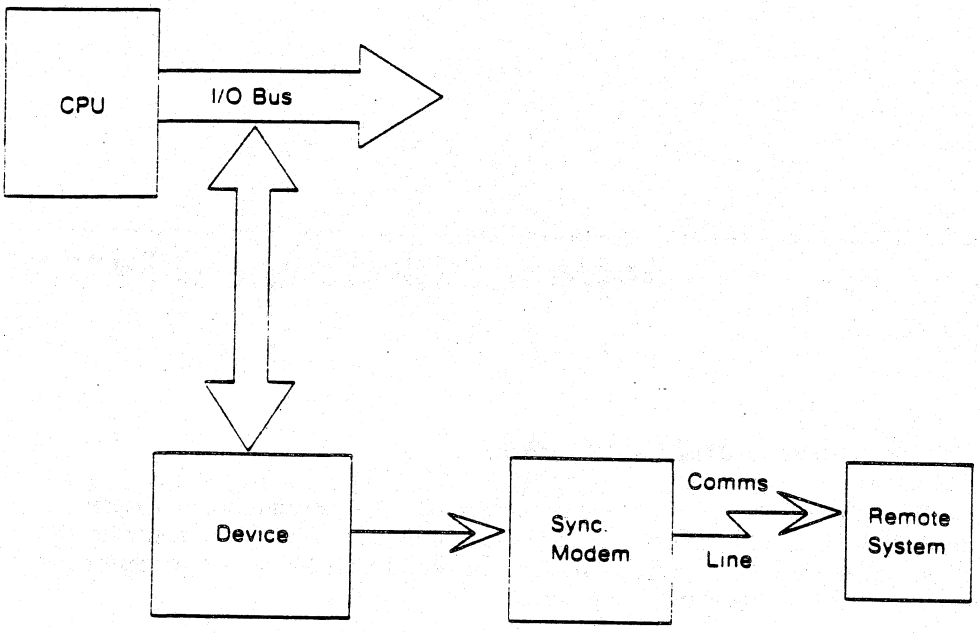
Note that extra information on using non-DDCMP protocols is in Section 3.6.

The DSV11 driver provides:

- A point-to-point operating mode in which the DSV11 is connected to another communications controller also operating in point-to-point mode.
- Asynchronous System Traps (ASTs) for transmitting attention conditions to your process.
- Full- and half-duplex operation (only full-duplex operation is available with HDLC).
- Multiple read and write buffers for transmitting and receiving data.
- Modem control. Appendix B contains state transition diagrams for modem control. The state transitions for modem control in full-duplex mode (Figure B–1) and in half-duplex mode (Figure B–2) are illustrated.

Note that the DSV11 driver does not provide DMC11 compatibility mode. Figure 1–1 shows a typical DSV11 configuration.

**Figure 1–1 Typical DSV11 Configuration**




### 1.1.1 System Quotas

The DSV11 driver transmits data using buffered I/O operations. Therefore, all transmit operations are limited by the buffered I/O quota of the calling process.

The quotas for the receive buffer are the process's buffered I/O quota and buffered I/O byte count quota.

Note that the reception of data can demand a large number of ASTs. Ensure that your AST limit (ASTLM) is sufficient to cope with this demand.

### 1.1.2 MicroVAX System Power Failure

Once power returns after a MicroVAX system power failure, you must restart the MicroVAX and the DSV11 driver to resume communications.

# 2

# Installing the DSV11 Driver

## 2.1 Installation Information

The DSV11 driver is supplied on the following media:

- 1 x RX50 flexible disk
- 1 x TK50 tape cassette

You can mount the distribution media on any RX50 or TK50 device you choose.

The DSV11 driver kit provides on-line release notes. To see these release notes, you can:

- Display or print the release notes by executing only the first 10 steps of the installation procedure (in Section 2.2).
- Display or print the release notes as part of the DSV11 driver installation procedure. This is described in Step 10 in Section 2.2.
- Display or print the file SYS$HELP:SJ010.RELEASE_NOTES at any time after the DSV11 driver is installed.

You should ensure that you have a minimum of 300 free blocks for the DSV11 driver installation on the target system disk (see Table 2–1 for details).

A driver's code and associated control blocks are loaded into nonpaged pool. The DSV11 driver requires your system to have 40K bytes free in nonpaged pool (this does not include the nonpaged pool for your application buffer requirements). You should allow 2K bytes for the Unit Control Block (UCB) for each DSV11 device. Also allow 2.5K bytes for the CMD blocks.

You should adjust the SYSGEN parameters that control the allocation and deallocation of nonpaged dynamic memory before running the DSV11 driver. If this is not done, there may be a heavy impact on the CPU (particularly at high packet rates). Appendix C describes the SYSGEN parameters and the recommended values to use.

The Extended Initialisation block requires two pages (1.0K bytes) of memory. This space must be:

- Physically-contiguous
- Available when the first DSV11 device is connected during your DSV11 driver program

Allocate this space using the SYSGEN parameter SPTREQ. Because this memory is mapped in the Q22-bus space, two map registers are also required.

To set SYSGEN values on your system, edit the MODPARAMS.DAT file and run the AUTOGEN utility. See Chapter 11 of the *Guide to VAX/VMS System Management and Daily Operations* for details of this procedure.

Table 2–1 lists the DSV11 driver files, their size, where they are located after the installation, and a brief description of their contents.

**Table 2–1 Installed Device Driver Files**

| Filename | Location | Block Size | Description |
|---|---|---|---|
| SJDRIVER.EXE | SYS$SYSTEM | 80 | The device driver |
| SJ010.RELEASE_NOTES | SYS$HELP | 31 | DSV11 driver release notes |
| SJDRIVER.ULD | SYS$SYSTEM | 90 | DSV11 firmware |
| SYNC$ACP.EXE | SYS$SYSTEM | 6 | DSV11 firmware loader |
| SJSTARTUP.COM | SYS$MANAGER | 4 | DSV11 ACP startup DCL |

## 2.2 Installation Procedure

Installing the DSV11 driver takes approximately two minutes and is described below. For more details of software installation on VAX/VMS systems, see the *Guide to VAX/VMS Software Installation*: Chapter 5 describes the VMSINSTAL procedure.

Note that the DSV11 firmware is soft-loaded at the end of the installation procedure and after you have run VMSINSTAL. To load the firmware, follow the instructions displayed during VMSINSTAL (as described in step 15 of the installation procedure).

To install the driver, follow the steps below. Where applicable, default values are indicated in square brackets after the prompt. To accept the default, press <RET>.

1. Ensure that you have a good backup copy of your system disk because the system disk is written to during the installation.
2. During the installation, DIGITAL recommends that you:
   - Log into the system account
   - Have no other processes running
   - Do not run DECnet
3. Load the media onto your selected device.
4. Enter these commands to begin the installation:

```
$ SET DEF SYSSUPDATE
$ @VMSINSTAL
```

   Alternatively, you can substitute the following commands:

```
$ SET DEF SYSSUPDATE
$ @VMSINSTAL product devname OPTIONS N
```

   where *product* is the name of the product you are installing (the product name is described in Step 8) and *devname* the name of the device where the distribution media is mounted (for example, DUA0:). OPTIONS N makes the release notes available, as described in Step 10.

   If you use the second form of the VMSINSTAL command line, Steps 7 and 8 of the installation procedure will not be executed.

   The installation now proceeds automatically and the system will prompt you for any information that is required.

5. If you are not logged in to the SYSTEM account, or processes are still running, warning messages will be displayed and you will be asked:

```
* Do you want to continue anyway [NO]?
```

   If continuing will cause problems, press <RET> to cancel the installation. Enter YES to continue the installation.

6. The next prompt is:

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

   If you are not satisfied with the backup, enter NO to cancel the installation. If you are satisfied with the backup, press <RET> to continue the installation.

7. You will now be prompted for the device where the distribution media is mounted:

```
* Where will the distribution volumes be mounted:
```

   Enter the name of the device (for example, DUA0:).

8. The system will ask:

```
Enter the products to be processed from the first distribution
volume set.
* Products:
```

Since there is only one product on the distribution volume, enter the wildcard character *, or use the product name SJ*mmn*, where *mm* is the major version number (2 digits), and *n* is the update number (1 digit). For example, for version 1.0 enter SJ010.

9. VMSINSTAL will now ask:

```
Please mount the first volume of the set on devname.
* Are you ready?
```

where *devname* is the name of the device you specified in Step 7.

If you have mounted the distribution volume, enter YES. If you have not mounted the distribution volume, mount it now and then enter YES.

10. If you ran VMSINSTAL without option N selected, the installation procedure moves to Step 11. If you ran VMSINSTAL with option N selected, you will now be asked about displaying or printing the on-line release notes:

```
Release Notes Options:

        (1) Display release notes
        (2) Print release notes
        (3) Both

* Select option [3]:
```

If you select option 1, the release notes are displayed on your terminal. If you select option 2, you are prompted for a queue name:

```
* Queue name [SYSSPRINT]:
```

The release notes are queued to the SYSSPRINT queue if you press <RET>, or are queued to the print queue you name at the prompt.

If you select option 3, both option 1 and option 2 are executed.

After this, VMSINSTAL asks:

```
Do you want to continue with the installation [NO]?:
```

If you only want to access the release notes, press <RET> or enter NO to end the installation. Enter YES to continue the installation.

11. You will now be asked:

```
* Do you want to purge files replaced by this installation [YES]?
```

If you want to purge the files from a previous driver installation, press <RET> or enter YES. If you want to save these files, enter NO. If this is the first installation of the synchronous driver on your system, press <RET> to continue the installation.

When you have answered this question, the installation goes ahead and concludes with execution of the DSV11 driver Installation Verification Procedure (IVP).

12. When the installation is complete, the installation procedure displays:

```
Installation of SJ version completed at time
```

where *version* is the version of the DSV11 driver (for example, V1.0) and *time* is the time at which the DSV11 driver installed (for example, 12:36).

13. The installation procedure now asks:

```
Enter the products to be processed from the next distribution
volume set.
* Products:
```

Since there are no more products to be installed, enter <CTRL/Z> or the command EXIT to end the installation.

14. VMSINSTAL exits with the message:

```
VMSINSTAL procedure done at time
```

where *time* is the time at which VMSINSTAL exited (for example, 12:37).

When VMSINSTAL exits, the DCL prompt is displayed again. If VMSINSTAL exits in this way, the driver has installed successfully.

15. At this point, the DSV11 has no ROM-resident firmware. Instead, the file containing the firmware is part of the DSV11 driver kit you have just installed. You must now ensure that the firmware loads onto the DSV11 board. Without the firmware, the DSV11 will not work.

To load the firmware onto the board, include the following command in your system specific startup procedure:

```
$@SYS$MANAGER:SJSSTARTUP
```

This ensures that the firmware is loaded when your system starts (for example, after the reboot indicated in step 16). Make sure this command comes before the commands starting the layered products that use the DSV11.

You can also load the firmware yourself by entering the SJSSTARTUP command after the DSV11 driver is loaded and connected (as described in step 16). If you enter the command yourself, monitor the OPCOM messages generated during the running of the SJSSTARTUP procedure. These messages indicate whether the DSV11 starts correctly: if it does, the firmware has loaded successfully.

16. To use the DSV11 driver you have just installed, reboot the system. VMS automatically loads the DSV11 driver at boot-time if the DSV11 hardware is present. Please note the following points:

- If this is the first installation of a DSV11 driver on your system, you can use the SYSGEN commands LOAD and CONNECT to load the DSV11 driver without rebooting.

- If this installation replaces an existing version of the DSV11 driver, reboot your system. You can also use the SYSGEN command RELOAD to replace the DSV11 driver if it is not busy. If the DSV11 driver is busy, a warning message is issued.
- In a cluster environment, the DSV11 driver image is installed into the cluster-wide directory SYSSSYSTEM. You must reboot all cluster nodes that have DSV11 devices.

An example installation is shown in Section 2.2.1.

### 2.2.1 Example Installation

The following is the log of a DSV11 driver installation. The product name and device where the distribution media is mounted (DUA0:) are given on the VMSINSTAL command line. The user is prompted for how the release notes should be displayed because OPTIONS N is also selected on the VMSINSTAL command line. The release notes will be printed from the SYSSPRINT queue.

```
$ set def sys$update
$ @vmsinstal dsv11 dua0: options n

        VAX/VMS Software Product Installation Procedure V4.7

It is 19-APR-1988 at 15:57.
Enter a question mark (?) at any time for help.

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

The following products will be processed:
        DSV11 V1.0

        Beginning installation of DSV11 V1.0 at 15:57

%VMSINSTAL-I-RESTORE, Restoring product saveset A...

Release Notes Options:

        1.  Display release notes
        2.  Print release notes
        3.  Both 1 and 2

* Select option [3]: 2

* Queue name [SYS$PRINT]:

* Do you want to continue the installation [N]? y

%VMSINSTAL-I-RELMOVED, The products release notes have been
        successfully moved to SYS$HELP.

* Do you want to purge files replaced by this installation [YES]? n

        The DSV11 Synchronous Line driver is now being
        installed.
```

```
    After VMSINSTAL exits, please add the following
    command to your site-specific startup procedure
    before the commands that start any DIGITAL layered
    products that use the DSV11 device:

    $ @SYS$MANAGER:SYSTARTUP

    This command loads the DSV11 firmware into the
    DSV11 device.

    To use the driver you have just installed, reboot
    the system.  VMS automatically loads the driver at
    boot-time if the DSV11 hardware is present.
    Also note the following points:

    * If this is the first installation of a DSV11
      synchronous driver on your system, you can use
      the SYSGEN commands LOAD and CONNECT to load
      the driver without rebooting.

    * If this installation replaces an existing
      version of the DSV11 synchronous driver,
      reboot your system or (if the DSV11 driver is
      not busy) use the SYSGEN command RELOAD.

    * In a cluster environment, the driver image is
      installed into the cluster-wide directory
      SYS$SYSTEM.  You must reboot all nodes in the
      cluster with DSV11 devices.

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

  Beginning the SCPDRVER Installation Verification Procedure

%ANALYZE-I-ERRCPS, SYS$SYSROOT:[SYSEXE]SYCPDRVER.EXE;1       0 errors
%ANALYZE-I-ERRCRS, SYS$SYSROOT:[SYSEXE]SYNCSACP.EXE;1        0 errors
        Installation of SC V1.0 completed at 15:58

Enter the products to be processed from the next distribution
volume set.
* Products: EXIT

  VMSINSTAL procedure done at 15:58

$
```

## 2.3 Testing the Installation

As noted in Section 2.2, the DSV11 driver installation procedure automatically tests that the DSV11 driver has been correctly installed on your system. If the DSV11 hardware is already installed on your system, the DSV11 driver will be autoconfigured when you reboot the system.

You can also check that the DSV11 devices are present on your system by issuing the command:

```
$ SHOW DEVICE SC
```

When the DSV11 devices are present, the display reads:

```
Device                  Device          Error
Name                    Status          Count
SCA0:                   Online              0
SCA1:                   Online              0
```

If you receive this error message:

```
%SYSTEM-W-NOSUCHDEV, no such device available
```

it indicates that the DSV11 may not have been installed at the correct address. In this case, consult the *DSV11 Communications Option Installation Guide* for details of installing the DSV11 at the correct address, or consult your Field Service representative.

For further checks on your communications environment, consult the documentation of the communications product you intend to use with the DSV11 driver (for example, DECnet–VAX).

# 3
# DSV11 Driver Function Codes

## 3.1 Overview of I/O Operations

The DSV11 driver performs these basic functions:

- Read (see Section 3.2)
- Write (see Section 3.3)
- Set Mode (see Section 3.4)
- Set Characteristics (see Section 3.4)
- Sense Mode (see Section 3.5)

Table 3–1 lists these functions and their function codes. The key to the table is as follows:

L Logical

V Virtual

P Physical

(H) Only for half-duplex operations

**Table 3–1 DSV11 Driver I/O Functions**

| Function Code and Arguments | Type | Modifiers | Function |
|---|---|---|---|
| IO$_READLBLK<br>P1,P2 | L | IO$M_NOW | Read logical block |
| IO$_READVBLK<br>P1,P2 | V | IO$M_NOW | Read virtual block |
| IO$_READPBLK<br>P1,P2 | P | IO$M_NOW | Read physical block |
| IO$_WRITELBLK<br>P1,P2 | L | IO$M_LASTBLOCK (H) | Write logical block |
| IO$_WRITEVBLK<br>P1,P2 | V | IO$M_LASTBLOCK (H) | Write virtual block |
| IO$_WRITEPBLK<br>P1,P2,[P6] | P | IO$M_LASTBLOCK (H) | Write physical block |
| IO$_SETMODE<br>P1,[P2],P3 | L | IO$M_CTRL<br>IO$M_SHUTDOWN<br>IO$M_STARTUP<br>IO$M_ATTNAST | Set DSV11 driver characteristics and state for subsequent operations |
| IO$_SETCHAR<br>P1,[P2],P3,[P6] | P | IO$M_CTRL<br>IO$M_SHUTDOWN<br>IO$M_STARTUP<br>IO$M_ATTNAST | Set DSV11 driver characteristics and state for subsequent operations |
| IO$_SENSEMODE<br>P1,P2 | L | IO$M_CTRL<br>IO$M_RD_MODEM<br>IO$M_CLR_COUNT<br>IO$M_RD_COUNT | Sense DSV11 driver characteristics and return them in specified buffer(s) |
| IO$_CLEAN | L | None | For HDLC and SDLC, stops all outstanding transmits. For BISYNC stops all outstanding I/O operations. Not used with DDCMP |

Generally, the DSV11 driver does not differentiate between logical, virtual, and physical I/O functions. However, there is one exception:

- You must have the required privilege to request a physical or logical function (for physical functions, PHY_IO privilege; for logical functions, LOG_IO privilege).

## 3.2 Read

A Read function transfers incoming data into the buffer you specify.

VAX/VMS provides three function codes:

- IO$_READLBLK — read logical block
- IO$_READVBLK — read virtual block
- IO$_READPBLK — read physical block

The DSV11 driver buffers the received data and copies it to the buffer you specify.

The parameters for the three function codes are:

**P1**

The starting virtual address of the buffer to receive the data.

**P2**

The size of the buffer in bytes. For BISYNC operation this buffer must be large enough to contain the whole BISYNC frame, including header, trailer, and checksum (see Section 3.6.1). P2 must not be larger than the maximum Receive-message size (see Section 4.1 for how to find the maximum Receive-message size). If a larger message is received, a status of SS$_BUFFEROVF is returned in the I/O status block (IOSB).

The Read functions can take the modifier:

IO$M_NOW — complete the read operation immediately with a received message. If no message is available when IO$M_NOW is applied, a status of SS$_ENDOFFILE is returned in the IOSB.

## 3.3 Write

A Write function transfers data from the buffer you specify and transmits the data down the line.

VAX/VMS provides three function codes:

- IO$_WRITELBLK — write logical block
- IO$_WRITEVBLK — write virtual block

- IOS_WRITEPBLK — write physical block

The DSV11 driver buffers your data in a system buffer before transmitting it.

The parameters for the three function codes are:

**P1**

The starting virtual address of the buffer holding your data.

**P2**

The size (in bytes) of the buffer holding your data. For BISYNC operation this buffer must be large enough to contain the whole BISYNC frame, including header, trailer, and checksum (see Section 3.6.1). P2 must not be larger than the maximum Send-message size (see Section 4.1 for how to find the maximum Send-message size).

The Write functions can take the modifier:

IOSM_LASTBLOCK — turns off Request To Send (RTS) after the transmit is sent (only for half-duplex operations).

## 3.4 Set Mode and Set Characteristics

The Set Mode and Set Characteristics functions control DSV11 driver operations. Principally, the Set Mode and Set Characteristics functions are used to:

- Specify the protocol to be used
- Specify the line speed
- Specify full- or half-duplex operation
- Allocate buffers
- Specify message size
- Request an attention AST
- Specify loop-back mode
- Enable/disable the internal clock and set the clock speed

The functions that perform these and other tasks are described in Sections 3.4.1 to 3.4.3. Extra information on using these functions with non-DDCMP protocols is in Section 3.6. Additional information on using the DDCMP protocol is in Section 3.7.

VAX/VMS defines three types of Set Mode function:

- Set/Start Controller mode (see Section 3.4.1)
- Shutdown controller (see Section 3.4.2)
- Enable attention AST (see Section 3.4.3)

VAX/VMS provides two function codes:

- IO$_SETMODE — set mode (requires logical I/O privilege)
- IO$_SETCHAR — set characteristics (requires physical I/O privilege)

### 3.4.1 Set Controller Mode

This function sets and (optionally) starts the DSV11 driver.

VAX/VMS provides four combinations of function code and modifier:

- IO$_SETMODE!IOSM_CTRL — set DSV11 driver characteristics
- IO$_SETCHAR!IOSM_CTRL — set DSV11 driver characteristics
- IO$_SETMODE!IOSM_CTRL!IOSM_STARTUP — set DSV11 driver characteristics and start the DSV11 driver
- IO$_SETCHAR!IOSM_CTRL!IOSM_STARTUP — set DSV11 driver characteristics and start the DSV11 driver

If the modifier IOSM_STARTUP is specified, the DSV11 driver is started and the modem is enabled. If IOSM_STARTUP is not specified, the DSV11 driver characteristics are simply modified.

The parameters for the function codes are:

**P1**

The virtual address of a quadword characteristics buffer. For further information see Section 3.4.1.1.

**P2**

Optional. The address of a descriptor for an extended characteristics buffer. For further information see Section 3.4.1.2.

**P3**

Number of Receive-message blocks to allocate (sometimes referred to as the size of the 'common receive pool'). For further information see Section 3.4.1.3.

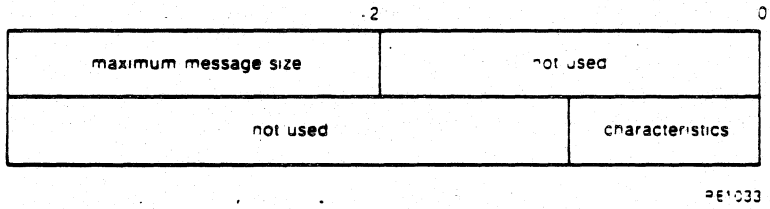
Note that if both the P1 and P2 parameters are specified, the P2 parameter values supersede the P1 parameter values. The P2 parameter NMASC_PCLI_BFN (see Table 3–3) also supersedes any P3 parameter.

Parameters P1, P2, and P3 are described in more detail in Sections 3.4.1.1 to 3.4.1.3.

### 3.4.1.1 P1 Parameter

P1 is the virtual address of a quadword characteristics buffer. This parameter is ignored for HDLC, SDLC, and BISYNC operations. Figure 3–1 shows the format of this buffer.

**Figure 3–1 P1 Characteristics Buffer (Set Controller)**




The second word of the first longword ('maximum message size') holds the maximum length for transmitted and received messages.

The first word of the second longword ('characteristics') defines the operational mode of the DSV11 driver.

Table 3–2 lists the DSV11 driver characteristics that can be set in the second longword. The SXMDEF macro defines these values.

**Table 3–2 DSV11 Characteristics**

| Characteristic | Meaning |
|---|---|
| XMSM_CHR_LOOPB | Sets loop back mode |
| XMSM_CHR_HDLPX | Sets half-duplex operation |

### 3.4.1.2 P2 Parameter

P2 is optional. It is the address of a descriptor that defines an extended characteristics buffer.

The extended characteristics buffer that P2 points to consists of a series of 6-byte entries. The first word contains the parameter identifier (ID) and the longword that follows contains a value that can be associated with that parameter ID. Figure 3–2 shows the format of this buffer.

**Figure 3–2 P2 Extended Characteristics Buffer**

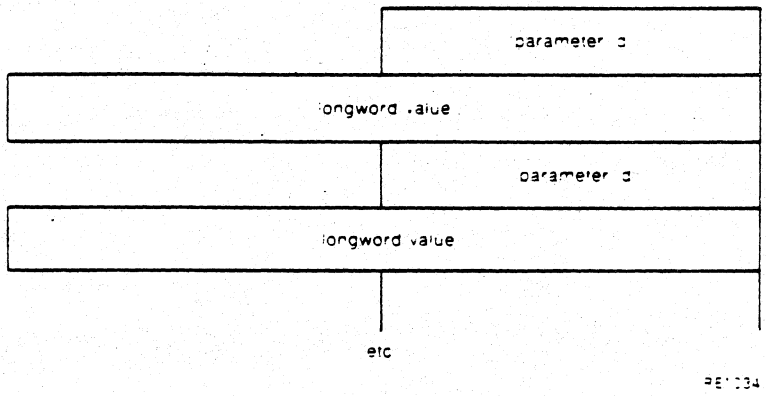



Table 3–3 shows the parameter IDs and possible values that can be specified in the P2 buffer (the notes referred to are at the end of the table). The SNMADEF macro defines these values.

**Table 3–3 P2 Extended Characteristics Values**

| Parameter ID | Meaning |
|---|---|
| NMASC_PCLI_PRO | Protocol mode. The following values can be specified: |

| Value | Meaning |
|---|---|
| NMASC_LINPR_POI | DDCMP point-to-point (default) |
| NMASC_LINPR_BISYNC | IBM bisynchronous protocol |
| NMASC_LINPR_LAPB | HDLC operation (LAPB) |
| NMASC_LINPR_LAPBE | HDLC operation (LAPBE) (see Note 1) |
| NMASC_LINPR_SDLC | SDLC bit stuff mode |

| | |
|---|---|
| NMASC_PCLI_DUP | Duplex mode (see Note 2 for defaults). The following values can be specified: |

| Value | Meaning |
|---|---|
| NMASC_DPX_FUL | Full-duplex |
| NMASC_DPX_HAL | Half-duplex (see Note 3) |

| | |
|---|---|
| NMASC_PCLI_CON | DSV11 mode. The following values can be specified: |

| Value | Meaning |
|---|---|
| NMASC_LINCN_NOR | Normal (default) |
| NMASC_LINCN_LOO | Loopback |

| | |
|---|---|
| NMASC_PCLI_BFN | Number of Receive buffers to preallocate (minimum = 1; for defaults, see Note 4). Must be provided here or as P3 argument (see Section 3.4.1.3). If included, supersedes the P3 argument. |

**Table 3–3 (Cont.) P2 Extended Characteristics Values**

| Parameter ID | Meaning |
|---|---|
| NMASC_PCLI_BUS | Maximum Transmit- and Receive-message length (for defaults and maximum values see Note 5). |
| NMASC_PCLI_NMS | Number of sync characters to precede message. The number used is protocol dependent (default = 8). |
| NMASC_PCLI_CODE | Character code used for IBM bisynchronous protocol. The following values can be specified: |

| Value | Meaning |
|---|---|
| NMASC_CODE_ASCII | ASCII character code |
| NMASC_CODE_EBCDIC | EBCDIC character code (default) |

NMASC_PCLI_NRZI — Data encoding technique. The following values can be specified:

| Value | Meaning |
|---|---|
| NMASC_STATE_OFF | RZI encoding (default) |
| NMASC_STATE_ON | NRZI encoding |

NMASC_PCLI_CLO — Controls generation of a clock signal. The following values can be specified (see Note 6):

| Value | Meaning |
|---|---|
| NMASC_LINCL_EXT | Clock signal disabled (default) |
| NMASC_LINCL_INT | Clock signal enabled |

**Table 3–3 (Cont.) P2 Extended Characteristics Values**

| Parameter ID | Meaning |
|---|---|
| NMA$C_PCLI_RTT | (DDCMP only) Retransmit timer for full-duplex point-to-point mode and selection timer. Specify value in milliseconds (default = 3000). |
| NMA$C_PCLI_LNS | Controls the speed of the clock signal enabled by NMA$C_PCLI_CLO (can also be used for timeout control: see Appendix C). The following values can be specified: |

| Value | Meaning |
|---|---|
| 0 | Clock is disabled |
| 9600 | Clock speed (hertz) |
| 19200 | (Default = 9600) |
| 38400 | |
| 72000 | |
| 128000 | |
| 256000 | |

**Notes:**

1. Because LAPBE can handle larger quantities of data, you may have to allocate more buffers for LAPBE operations than for LAPB operations.
2. The default duplex mode for each protocol is:

   DDCMP Full-duplex

   HDLC Full-duplex (no half-duplex mode with HDLC)

   SDLC Half-duplex

   BISYNC Half-duplex

3. In half-duplex mode, your program must signal the change from a TRANSMIT to a RECEIVE state. To signal this change, use the function code modifier IO$M_LASTBLOCK with the last IO$_WRITE call in a sequence.

   When the DSV11 driver is in half-duplex mode:

   - You can issue multiple IO$_READ calls. These will be accepted whatever the direction of the line at the time of issue and will not be aborted if the line changes to TRANSMIT.

- If there is no carrier from the far end, the first IO$_WRITE call you issue will place the line into the TRANSMIT state. Hence, Request To Send (RTS) will be raised and, when Clear To Send (CTS) is raised, the data will be transmitted.
- Use the modifier IO$M_LASTBLOCK with your final IO$_WRITE call to indicate the final piece of data in a transmit sequence. If the IO$_WRITE call includes IO$M_LASTBLOCK, this data (but no subsequent data) will be sent to the DSV11 driver for transmission.
- On completion of a transmission with the IO$M_LASTBLOCK modifier, RTS is dropped. The line direction is left indeterminate until there is an indication from the DSV11 device that CTS has been dropped and Carrier Detect (DCD) has been raised. The line direction is then set to RECEIVE and will remain so until DCD is dropped. However, if a transmit is queued before DCD is detected, the line direction is again set to TRANSMIT.
- Any transmits are queued until DCD is dropped. RTS is then raised and the transmits queued for transmission until a transmit with the IO$M_LASTBLOCK modifier comes through.

4. Default number of buffers allocated:

   DDCMP 4

   HDLC 6

   SDLC 4

   BISYNC 2

5. Default message length (in bytes):

   DDCMP 576

   HDLC 128

   SDLC 280

   BISYNC 280

   Maximum message length (in bytes):

   DDCMP 4096

   HDLC 4106

   SDLC 4106

   BISYNC 4106

6. DIGITAL recommends that NMASC_PCLI_CLO be left at its default value. Set the line speed using the NMASC_PCLI_LNS parameter only when NMAS_PCLI_CLO sets the internal clock. Setting the line speed with NMAS_PCLI_LNS when NMAS_PCLI_CLO sets an external clock has no effect on the line speed used by the DSV11 driver. Note that there is no method of obtaining the current value of the line speed parameter.

#### 3.4.1.3 P3 Parameter

P3 is the number of Receive-message blocks you are allocating for incoming data; that is, the size of the 'common receive pool' (see NMASC_PCLI_BFN Parameter ID in Table 3–3). This parameter is ignored for HDLC, SDLC, and BISYNC operations.

### 3.4.2 Shutdown Controller

This function ends DSV11 driver operations and halts the protocol and the line. To restart the DSV11 driver, issue a IO$_SETMODE!IO$M_CTRL!IO$M_STARTUP or IO$_SETCHAR!IO$M_CTRL!IO$M_STARTUP request (see Section 3.4.1).

Note that the defaults are not reset on shutdown, but only on DEASSIGN. The DSV11 driver uses its previous settings on a restart after a shutdown. To change the settings after a shutdown, use the P2 parameter as described in Section 3.4.1.2.

VAX/VMS provides two combinations of function code and modifier:

- IO$_SETMODE!IO$M_CTRL!IO$M_SHUTDOWN — shutdown DSV11 driver
- IO$_SETCHAR!IO$M_CTRL!IO$M_SHUTDOWN — shutdown DSV11 driver

### 3.4.3 Enable Attention AST

This function requests that an attention AST is delivered to the requesting process after one of the following events:

- THE DSV11 driver has set or cleared any of the DSV11 device and line status bits (see Table 4–3).
- The DSV11 driver has set or cleared a DSV11 error summary bit (see Table 4–4).
- Data has arrived and there is no waiting IO$_READ request.

All outstanding attention ASTs are delivered after one of these events.

You may use the Enable Attention AST function at any time after the line is started, regardless of the condition of the DSV11 device and line status bits.

VAX/VMS provides two combinations of function code and modifier:

- IO$_SETMODE!IO$M_ATTNAST — enable attention AST

- IO$_SETCHAR!IO$M_ATTNAST — enable attention AST

The parameters for the two function codes are:

**P1**

The address of an AST service routine (or 0 to disable ASTs).

**P2**

Ignored.

**P3**

Access mode to deliver AST (0 to 3, corresponding to the VMS access mode chosen). If you specify a more privileged access mode than the current access mode of the calling process, the AST is delivered at the current access mode. Otherwise, the AST is delivered at the access mode you have specified.

After an AST occurs, it must be reenabled by another Enable Attention AST function before an AST can occur again. Note that the AST quota (ASTLM) for your process limits how many ASTs can be requested.

The AST service routine is called and given an argument list. The first argument is the value in the IOSB's second longword (see Chapter 5). Ensure that argument lists for any remaining entries are preserved.

## 3.5 Sense Mode

The Sense Mode function returns the DSV11 driver characteristics (excluding the line speed characteristic) in the specified buffer(s).

VAX/VMS provides one function code:

- IO$_SENSEMODE!IO$M_CTRL — read DSV11 driver characteristics

The parameters for the function code are:

**P1**

Optional. The address of a two-longword buffer for DSV11 driver characteristics. See Figure 3–1.

**P2**

Optional. The address of a descriptor that defines a DSV11 driver extended characteristics buffer. See Figure 3–2.

If all the characteristics cannot be stored in the buffer you specify, the IOSB returns:

- SS$_BUFFEROVF in the first word
- The size (in bytes) of the extended characteristics buffer in the second word

Note that the size of the buffer returned may differ from the size of the buffer you specified. This happens when the sizes of the characteristics definitions do not fit exactly into the buffer. For example, if the DSV11 driver has 8 6-byte characteristics to return (total 48 bytes) and the buffer is 20 bytes long, only 3 characteristics will be returned (total 18 bytes).

For a description of the IOSB, see Chapter 5.

## 3.6 Using Non–DDCMP Protocols

The HDLC, SDLC, and BISYNC protocols do not have the concept of line and circuit. Therefore, only SQIO requests including the modifier IO$M_CTRL are allowed. VMS does not acknowledge the characteristics set in the P1 buffer for this mode of operation.

### 3.6.1 BISYNC

You must construct and pass a complete BISYNC frame to the DSV11 driver when in BISYNC mode. This frame must include all framing and control characters (for example, the DLE, STX, ETB, and ETX characters). You must also leave space in the frame at the correct points for the DSV11 driver to insert checksums (two bytes for each CRC).

### 3.6.2 The IO$_CLEAN Function

For HDLC and SDLC, an IO$_CLEAN function stops all outstanding Transmits. For BISYNC, an IO$_CLEAN function stops all outstanding I/O operations. In both cases, the status return is SS$_ABORT. Note that the modem registers are not cleared by IO$_CLEAN.

IO$_CLEAN is not used with DDCMP.

## 3.7 Using the DDCMP Protocol

After you have set up the controller mode using the IOS_SETMODE!IOSM_CTRL!IOSM_STARTUP function code (as described in Section 3.4.1) you need to set and start the DDCMP protocol. Use the Set DDCMP mode function to set and start the DDCMP protocol.

Four combinations of function code and modifier are provided:

- IOS_SETMODE — modify DDCMP characteristics
- IOS_SETCHAR — modify DDCMP characteristics
- IOS_SETMODE!IOSM_STARTUP — start DDCMP protocol
- IOS_SETCHAR!IOSM_STARTUP — start DDCMP protocol

These codes take the following arguments:

**P1**

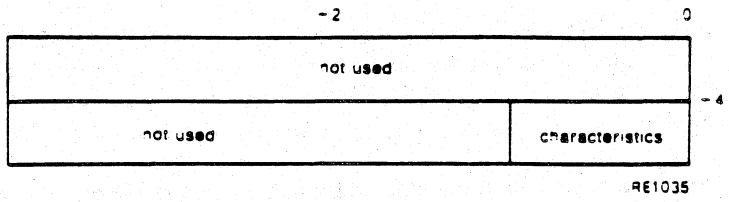
The virtual address of a quadword characteristics buffer (optional).

**P2**

The address of a descriptor for an extended characteristics buffer (optional).

The P1 buffer has the structure shown in Figure 3–3.

**Figure 3–3 P1 Characteristics Buffer (Set DDCMP)**




The following characteristic can be set in the second longword:

XMSV_CHR_MOP — set DDCMP to maintenance mode

The P2 buffer consists of a series of 6-byte entries. The first word contains the parameter identifier (ID), and the longword that follows contains one of the values that can be associated with the parameter ID. Figure 3–2 shows the format for this buffer.

Table 3–4 lists the parameter ID and values that can be specified in the P2 buffer.

**Table 3–4 P2 Extended Characteristics Values**

| Parameter ID | Meaning |
|---|---|
| NMASC_PCCI_MTR | Maximum number of data messages in a row transmitted before deselecting (default = 4) |
| NMASC_PCCI_MST | DDCMP maintenance mode. The following values can be specified: |

| Value | Meaning |
|---|---|
| NMASC_STATE_OFF | DDCMP maintenance mode disabled (default) |
| NMASC_STATE_ON | DDCMP maintenance mode enabled |

If both P1 and P2 characteristics are specified, the P2 characteristics supersede the P1 characteristics. For example, if P1 specifies XMSM_CHR_MOP and P2 specifies NMASC_PCCI_MST with a value of NMASC_STATE_OFF, DDCMP is in normal data mode.

On receipt of the IO$_SETMODE!IO$M_STARTUP QIO request, the DSV11 driver starts the DDCMP protocol.

Section 3.7.1 describes how to shutdown the DDCMP protocol initiated by the IO$_SETMODE!IO$M_STARTUP QIO request.

### 3.7.1 Shutdown DDCMP

For the DSV11 driver, this function halts the DDCMP protocol. The attached device cannot be used for data transfer until DDCMP is restarted.

Two combinations of function code and modifier are provided:

- IO$_SETMODE!IO$M_SHUTDOWN — shutdown DDCMP
- IO$_SETCHAR!IO$M_SHUTDOWN — shutdown DDCMP

These codes take no arguments.

## 3.8 Modem Control

There are two modes of modem control with the DSV11 driver:

1. Full-duplex
2. Half-duplex

Section 3.8.1 contains general information on modem control. Section 3.8.2 describes full-duplex modem control and Section 3.8.3 describes half-duplex modem control. Appendix B contains state transition diagrams for the modem during full- and half-duplex operation.

### 3.8.1 General Information

The DSV11 driver:

- Must have Data Set Ready (DSR) and Carrier Detect (DCD) from the modem.
- Requires that Clear To Send (CTS) is dropped by the modem if the DSV11 driver drops Request To Send (RTS).
- Will queue data to the DSV11 device during line startup (when the DSV11 device is waiting for a DCD, CTS, or DSR signal). The data is transmitted after a signal is received.

Care should also be taken over the following points when operating in full- or half-duplex modes:

- Full-duplex operation requires that the DCD, the DSR, and the CTS signals are present. Data will be transmitted only if these signals are present when the DSV11 driver sets Data Terminal Ready (DTR) and RTS.
- Half-duplex operation requires that DSR is present at all times. CTS must be raised by the modem when the DSV11 driver raises DTR, and CTS must be dropped by the modem when the DSV11 driver drops RTS. For transmission to take place, DCD must be dropped by the modem when no data is being received. This usually occurs because the remote machine drops RTS when it has finished sending data.
- HDLC operation (full-duplex only) requires that DSR and CTS are high. DCD must also be present, except for absences of less than 15 seconds.

### 3.8.2 Full–duplex

This is the default for HDLC and DDCMP. All modem bits must be provided by the modem. DTR is set when this mode is entered. RTS is kept on while it is possible to send data. This mode can be used for dial-up access because DTR is dropped when DCD, CTS, or DSR is lost.

DTR is cleared when a disconnect is detected. It remains clear for a minimum of 250 ms and for up to 5 seconds until DSR goes off. After this DTR is set again, waiting for DSR to indicate a new call.

If DSR is seen in the idle state, the driver sets RTS and waits for up to 5 seconds for both DCD and CTS. If the 5 seconds run out, the call is cleared. Once the call is accepted, a change in CTS or DSR will clear the call immediately.

If DCD is lost, then a timer of 15 seconds is started. If the 15 seconds run out before DCD returns, the call is cleared.

When a call is cleared, the XMSM_STS_DISC bit is set in the device status. This is not treated as a fatal error (except for DDCMP operation) and, unless your program takes some other action, the call can be reestablished (XMSM_STS_DISC will be cleared).

The state transitions for full-duplex mode are shown in Figure B–1 in Appendix B.

### 3.8.3 Half–duplex

This is the default for BISYNC and SDLC. All protocols operate in the same half-duplex mode (HDLC only operates in full-duplex mode). RTS is used in this mode when the driver is transmitting data. The device is set to idle MARK.

See Section B.2 for how DTR, DSR, RTS, and CTS are used. DCD is used to indicate that reception is possible. When DCD is set the driver will not set RTS.

The state transitions for half-duplex mode are shown in Figure B–2 in Appendix B.

## 3.9 Modem Status

The Read Modem Status function reads the DSV11 device modem status register.

VAX/VMS provides two combinations of function code and modifier:

- IO$_SENSEMODE!IO$M_CTRL!IO$M_RD_MODEM — read modem status
- IO$_SENSECHAR!IO$M_CTRL!IO$M_RD_MODEM — read modem status

There is one parameter for the function codes:

**P1**

The address of a longword buffer which stores the modem status. One or more of the bits described in Table 3–5 can be set in this buffer.

**Table 3–5 Modem Status Bits**

| Bit | Meaning |
|---|---|
| XMSV_MDM_CARRDET | Receiver is active (carrier detected) |
| XMSV_MDM_CTS | Data can be transmitted (CTS) |
| XMSV_MDM_DSR | Modem is in service (DSR) |
| XMSV_MDM_RTS | Request to send data (RTS) |
| XMSV_MDM_DTR | Modem is available and on-line (DTR) |
| XMSV_MDM_RING | Modem has just been dialed up (INDICATE) |

# 4
# Getting DSV11 Information

## 4.1 How to Get Information

To get information about DSV11 characteristics use the Get Device/Volume Information (SGETDVI) system service. For information on SGETDVI, see the *VAX/VMS System Services Reference Manual*.

For the DSV11, SGETDVI returns the following information:

- DSV11 device characteristics
- DSV11 device class
- DSV11 device type
- Maximum message size
- DSV11 status
- Line status
- Error returns

To get DSV11 device characteristics, call SGETDVI with item code DVIS_DEVCHAR. Table 4–1 lists these characteristics, which are defined by the SDEVDEF macro.

**Table 4–1 DSV11 Device Characteristics**

| Static Bits (always set) | Meaning |
|---|---|
| DEVSM_AVL | Device available. Set when UCB (Unit Control Block) initialized |
| DEVSM_IDV | Input device |
| DEVSM_NET | Network device. Set for terminal port if it is a network device |
| DEVSM_ODV | Output device |

To get the DSV11's device class, call SGETDVI with item code DVIS_DEVCLASS. The DSV11's device class is DCS_SCOM.

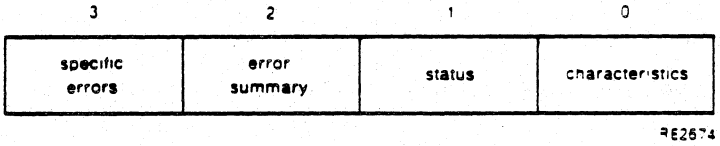
To get the DSV11's device type, call SGETDVI with item code DVIS_DEVTYPE. The DSV11's device type is DTS_DSV11.

The SDCDEF macro defines the device class and device type names.

To get the maximum message size, call SGETDVI with item code DVIS_DEVBUFSIZ. The maximum message size is the maximum Send- or Receive-message size you have defined for the DSV11 driver. Note that, on modem-controlled lines, transmission errors increase as message size increases.

To get DSV11 status and error information, call SGETDVI with item code DVIS_DEVDEPEND. SGETDVI returns a longword containing this information. The format of the longword is shown in Figure 4–1.

**Figure 4–1 Longword Returned by $GETDVI**




The longword contains:

- DSV11 driver characteristics (byte 0)
- DSV11 device and line status (byte 1)
- DSV11 error summary (byte 2)

- DSV11 specific error(s) (byte 3)

The contents of these fields are described in Sections 4.1.1 to 4.1.4.

### 4.1.1 DSV11 Driver Characteristics

The DSV11 driver characteristic bits govern the DDCMP operating mode. These bits are defined by the SXMDEF macro and can be set using a Set Mode function (see Section 3.4.1) or read by a Sense Mode function (see Section 3.5).

Table 4–2 lists the values and meanings of the DSV11 driver characteristics.

**Table 4–2 DSV11 Driver Characteristics**

| Characteristic | Meaning |
|---|---|
| XMSM_CHR_HDPLX | Sets half-duplex operation |
| XMSM_CHR_LOOPB | Sets loop-back mode |
| XMSM_CHR_MOP | DDCMP maintenance mode |

### 4.1.2 DSV11 Device and Line Status

These bits show the status of the DSV11 device and of the line. Set or clear these bits only when the DSV11 and the circuit are inactive.

Table 4–3 lists the status values and their meanings. The values are defined by the SXMDEF macro.

**Table 4–3 DSV11 Device and Line Status**

| Status | Meaning |
|---|---|
| XMSM_STS_ACTIVE | DSV11 device and selected protocol are active (indicates establishment of a link to the remote device only in full-duplex mode) |
| XMSM_STS_BUFFAIL | Receive buffer allocation failed |
| XMSM_STS_DISC | Modem disconnected. This bit is returned in the field IRPSL_IOST2 if the DSV11 driver has detected an incorrect modem status (returns a fatal error with DDCMP) |

### 4.1.3 DSV11 Error Summary

The DSV11 error summary bits are set when an error occurs. They are read-only bits. If the error is fatal the DSV11 shuts down.

Table 4–4 lists the error values and their meanings.

**Table 4–4 DSV11 Error Summary**

| Error Summary Bit | Meaning |
|---|---|
| XMSM_ERR_FATAL | Hardware or software error occurred on DSV11 |
| XMSM_ERR_THRESH | Receive, Transmit, or Select threshold errors |
| XMSM_ERR_LOST | Data lost because longer message received than the specified maximum message size |
| XMSM_ERR_MAINT | DDCMP maintenance message received |
| XMSM_ERR_START | DDCMP start message received |
| XMSM_ERR_TRIB | Hardware or software error occurred on circuit |

### 4.1.4 DSV11 Specific Errors

The specific error bits indicate the precise error. Table 4–5 lists the errors and the error codes.

**Table 4–5 DSV11 Errors**

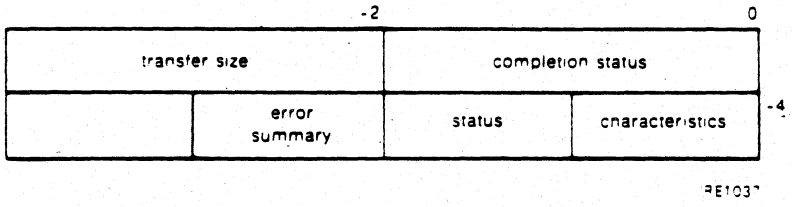
| Code Set | Meaning |
|---|---|
| XMSM_ERR_DATACHK | Software error |
| XMSM_ERR_FATAL | Severe error requiring line shutdown |
| XMSM_ERR_LOST | Buffer too small |
| XMSM_ERR_MAINT | Maint received in Run state |
| XMSM_ERR_START | Start received in Maint state |
| XMSM_ERR_START | Start received in Run state |
| XMSM_ERR_THRESH | Receive threshold error |
| XMSM_ERR_THRESH | Select threshold error |
| XMSM_ERR_THRESH | Transmit threshold error |
| XMSM_STS_DISC | Modem disconnect |
| None | Maint received in Halt state |
| None | Ring detect |

# 5
# I/O Status Block

The format of the I/O status block (IOSB) is shown in Figure 5–1. The format of an IOSB reporting an invalid SET MODE or SET CHAR parameter is shown in Figure 5–2.

See Appendix A for a list of the completion status returns. The *VAX/VMS System Messages and Recovery Procedures Reference Manual* provides explanations and suggested user actions for these returns.

**Figure 5–1 IOSB Contents**




Besides the completion status, the first longword of the IOSB returns one of two values:

- The size (in bytes) of the data transfer
- The size (in bytes) of the extended characteristics buffer returned by a Sense Mode function

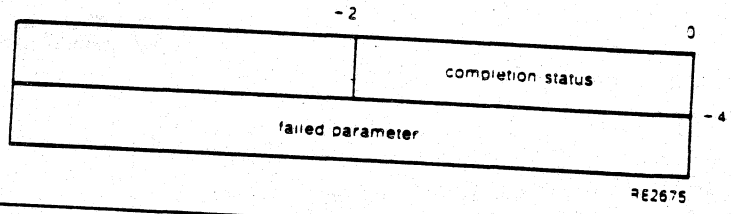
The second longword of the IOSB returns three values:

- The DSV11 driver characteristics (see Table 4–2)

- The DSV11 device and line status (see Table 4–3)
- The DSV11 error summary (see Table 4–4)

When the IOSB reports an invalid SET MODE or SET CHAR parameter, the format of the IOSB is as shown in Figure 5–2.

**Figure 5–2 IOSB Reporting Invalid Parameter**




The first word of the IOSB returns the completion status.

The second longword of the IOSB returns the name of the invalid parameter (as defined by the SNMADEF macro, and listed in Table 3–3).

# A

# I/O Function Codes

## A.1 Introduction

This appendix lists the function codes and function modifiers defined in the SIODEF macro. The functions grouped in the left-hand column take any of the arguments grouped in the right-hand column.

## A.2 DSV11 Function Codes

**Table A–1 DSV11 Function Codes**

| Functions | Arguments |
|---|---|
| IO$_READLBLK[IO$M_NOW]<br>IO$_READVBLK[IO$M_NOW]<br>IO$_READPBLK[IO$M_NOW]<br>IO$_WRITELBLK[IO$M_LASTBLOCK]<br>IO$_WRITEVBLK[IO$M_LASTBLOCK]<br>IO$_WRITEPBLK[IO$M_LASTBLOCK] | P1—buffer address<br>P2—buffer size |
| IO$_SETMODE<br>IO$_SETCHAR<br>IO$_SETMODE!IO$M_STARTUP<br>IO$_SETCHAR!IO$M_STARTUP<br>IO$_SETMODE!IO$M_CTRL<br>IO$_SETCHAR!IO$M_CTRL<br>IO$_SETMODE!IO$M_CTRL!IO$M_STARTUP<br>IO$_SETCHAR!IO$M_CTRL!IO$M_STARTUP<br>IO$_SETMODE!IO$M_SHUTDOWN<br>IO$_SETCHAR!IO$M_SHUTDOWN<br>IO$_SETMODE!IO$M_CTRL!IO$M_SHUTDOWN<br>IO$_SETCHAR!IO$M_CTRL!IO$M_SHUTDOWN | P1—optional. Characteristics buffer address<br>P2—optional. Extended characteristics buffer descriptor address<br>P3—optional. Number of receive message blocks |
| IO$_SETMODE!IO$M_ATTNAST<br>IO$_SETCHAR!IO$M_ATTNAST | P1—AST service routine address (zero disables ASTs)<br>P2—(ignored)<br>P3—access mode to deliver AST |
| IO$_SENSEMODE!IO$M_RD_MODEM<br>IO$_SENSEMODE!IO$M_CLR_COUNT<br>IO$_SENSEMODE!IO$M_RD_COUNT | P1—address of modem status buffer |
| IO$_SENSEMODE<br>IO$_SENSEMODE!IO$M_CTRL | P1—optional. Characteristics buffer address<br>P2—optional. Extended characteristics buffer descriptor address |
| IO$_SENSEMODE!IO$M_CTRL!IO$M_RD_MODEM | P1—status address of modem |
| IO$_CLEAN | (none) |

## A.3 QIO Status Returns

**Table A–2 DSV11 QIO Status Returns**

| QIO Status Returns | | |
|---|---|---|
| SS$_ABORT | SS$_BADPARAM | SS$_BUFFEROVF |
| SS$_CANCEL | SS$_DEVACTIVE | SS$_DEVICEFULL |
| SS$_DEVINACT | SS$_DEVOFFLINE | SS$_ENDOFFILE |
| SS$_INSFMEM | SS$_NORMAL | SS$_EXQUOTA |

For more information on these returns, including recovery action, refer to the *VAX/VMS System Messages and Recovery Procedures Manual*.

# B
# Modem Control State Transitions

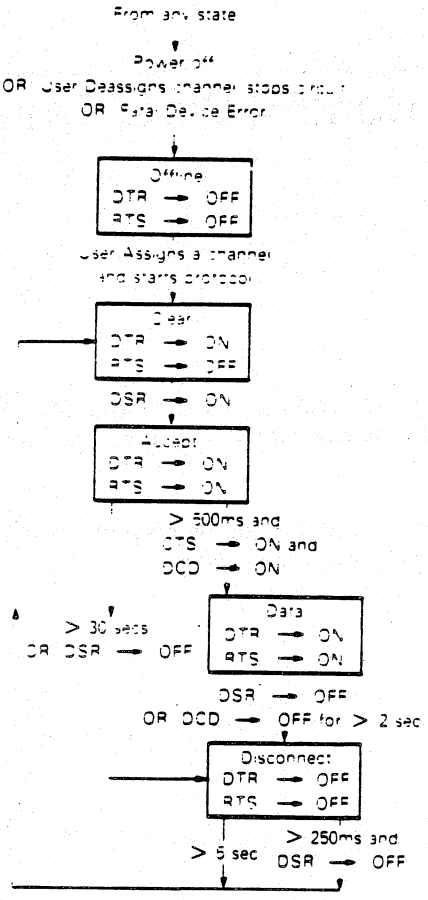
## B.1 Introduction

Figure B–1 shows the modem control state transitions when the DSV11 driver is in full-duplex mode. Figure B–2 shows the modem control state transitions when the DSV11 driver is in half-duplex mode.

The current state is named at the top of each box in the diagram. A descriptive list of the states is included on the right-hand side of the diagram.

## B.2 State Transition Diagrams

### Figure B–1 Full–duplex Modem Control




State Descriptions:

OFFLINE
No channel is assigned to the device or an error is detected

CLEAR
A channel is assigned to the device and the protocol started

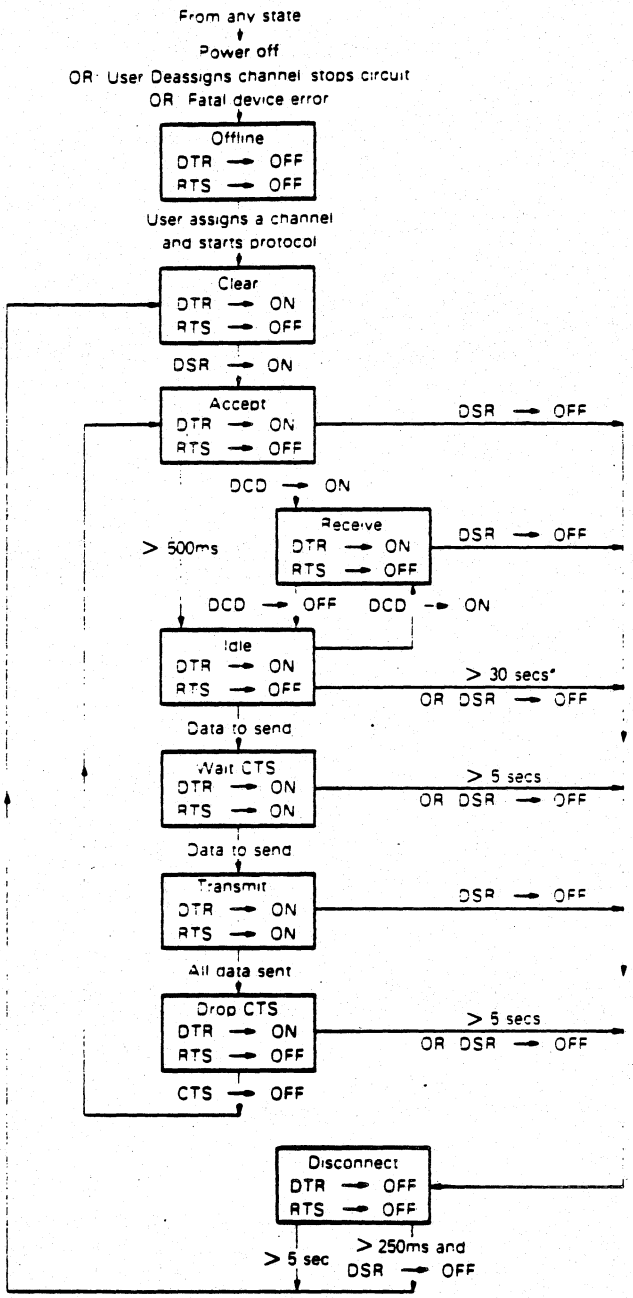
ACCEPT
The DSV11 is setting up the connection

DATA
Full-duplex data transfer can take place

DISCONNECT
The DSV11 is disconnecting the ch

RE-1208

## Figure B–2 Half–duplex Modem Control




State Descriptions:

OFFLINE
No channel is assigned to the device or an error is detected

CLEAR
A channel is assigned to the device and the protocol started

ACCEPT
The DSV11 is setting up the connection

RECEIVE
Half-duplex reception can take place

IDLE
Waiting to receive or transmit

WAIT CTS
Waiting for CTS

TRANSMIT
Half-duplex transmission can take place

DROP CTS
Dropping CTS

DISCONNECT
The DSV11 is disconnecting the link

RE1209

# C
# Tuning Your System

## C.1 Allocating and De–allocating Dynamic Memory

VMS organizes nonpaged pool into several lists of pre-formed buffers (called a 'lookaside' list) and an area of general nonpaged pool. Allocating nonpaged pool from a lookaside list requires little CPU activity, whereas allocation from the general area can require intensive CPU activity.

A system running the DSV11 driver at high packet rates may be allocating buffers from nonpaged pool. To minimize the CPU impact, you should ensure that buffers are allocated from a lookaside list. Do this by adjusting the DSV11 buffer size, or by adjusting the following SYSGEN parameters:

- SRPMIN
- SRPSIZE
- LRPMIN
- LRPSIZE

Follow this procedure to tune your use of nonpaged pool:

1. Find the DSV11 buffer size using the command SHOW DEVICE/FULL (or check the value you have specified in the NMASC_PCLI_BUS parameter to a IOS_SETMODE QIO).
2. Calculate the buffer size the DSV11 driver uses internally by adding 72 to the value found in Step 1 and rounding up to the nearest 16 byte boundary.
3. Enter SYSGEN and use the SHOW command to find the values of SRPMIN, SRPSIZE, LRPMIN, and LRPSIZE. The values for IRPMIN (97) and for IRPSIZE (196) are constants and cannot be changed.

4. Where feasible, adjust the buffer size so that it lies between one of the pairs of values SRPMIN/SRPSIZE, LRPMIN/LRPSIZE, or IRPMIN/IRPSIZE.

5. If it is not feasible to adjust the buffer size to fit between these values, use SYSGEN to change the system parameters as detailed in Table C–1. To set SYSGEN values on your system, edit the MODPARAMS.DAT file and run the AUTOGEN utility. See Chapter 11 of the *Guide to VAX/VMS System Management and Daily Operations* for details of this procedure.

**Table C–1 Adjusting SRP, LRP, and Buffer Size**

| Condition | Action |
|---|---|
| Buffer size < SRPMIN | Adjust SRPMIN to equal buffer size or adjust buffer size to equal SRPMIN |
| SRPMIN < buffer size < SRPSIZE | Buffers will be allocated from the SRP lookaside list |
| SRPSIZE < buffer size < IRPMIN | Adjust SRPSIZE to equal buffer size (IRPMIN cannot be adjusted) |
| IRPMIN < buffer size < IRPSIZE | Buffer will be allocated from the IRP lookaside list |
| IRPSIZE < buffer size < LRPMIN | Adjust LRPMIN to equal buffer size (IRPSIZE cannot be adjusted) |
| LRPMIN < buffer size < LRPSIZE | Buffers will be allocated from the LRP lookaside list |
| LRPSIZE < buffer size | If the disparity is not too great, consider increasing LRPSIZE to equal buffer size (note that large buffers inherently give fewer buffer allocations) |

**NOTE**

Taking these steps usually increases the dynamic memory used by the DSV11 driver. You should check the use of dynamic memory while the DSV11 driver is transferring data at peak rate. Use the MONITOR command MONITOR POOL for this purpose. Check that there are sufficient SRPs, LRPs, and IRPs, and that the overall pool is satisfactory.

# D
# Programming Example

## D.1 Introduction

This sample program shows the typical use of SQIO functions in DSV11 driver operations. The operations shown include starting the DSV11 driver, and transmitting and receiving data.

To run the following program on the DSV11 driver enter the initial DCL command:

```
$ ASSIGN SCA0: DEV
```

## D.2 Example Program

```
        .TITLE  EXAMPLE - DSV11  Device Driver Sample Program
        .IDENT  'X01'
        .LIBRARY
         SYS$LIBRARY:LIB
        $IODEF                          ; Define I O functions and modes
        $NMADEF                         ; Define Network Management symbols
        $XMDEF                          ; XMDRIVER definitions
;
;  Macro definitions
;
        .macro  type    string,tl                       ;
        store   <string>                                ;
        movl    #$$.tmpx,cmdcrab+rabSl_rbf              ;
        movw    #$$.tmpxl,cmdcrab+rabSw_rsz             ;
        $put    rab=cmdcrab                             ;
        blbs    r0,1                                    ;
        $exit_s                                         ;
1:                                                      ;
        .endm   type                                    ;
;
        .macro  store   string,pre
        .save
        .psect  $$$dev
        $$.tmpx=.
        pre
        .ascii  <string>
        $$.tmpxl=.-$$.tmpx
        .restore
        .endm   store

CMDOFAB:        $FAB    fac=put,fnm=sys$output:,- ; Output FAB
                        mrs=132,rat=cr,rfm=var
CMDORAB:        $RAB    ubf=cmdbuf,usz=cmdbsz,- ; Output RAB
                        fab=cmdofab
CMDBUF::        .BLKB   256                     ; Command buffer
CMDBSZ=         .-CMDBUF                        ; Buffer size
FACBUFDSC:      .LONG   CMDBSZ,CMDBUF           ; FAC buffer
                                                ; descriptor
FACLEN:         .BLKL   1                       ; FAC output buffer
                                                ; length
P2BUF::         .BLKB   6                       ; P2 buffer
P2BUFSZ=        .-P2BUF                         ; P2 buffer size
P2BUFDSC:       .LONG   P2BUFSZ,P1BUF           ; P2 buffer descriptor
P1BUF::         .BLKQ   1                       ; P1 buffer
```

```
PIBUFSZ=        .-PIBUF                         ; PI buffer size
CHNL::          .BLKL   1                       ; Channel number
IOSB::          .BLKL   2                       ; I/O status block
DEVDSC:         .ASCID 'DEV'                    ; Device to assign
QIOREQDSC:      .LONG   QIOREQSZ,QIOREQ         ; QIO request status
QIOREQ:         .ASCII  'QIO completion status = !XL '
                .ASCII  'IOSB1 = !XL, IOSB2 = !XL'
QIOREQSZ=       .-QIOREQ                        ; Size of QIO status
                                                ; report
XMTBUFLEN=256                                   ; Size of transmit
                                                ; buffer
XMTBUF:         .REPEAT XMTBUFLEN
                .BYTE   ^X93                    ; Transmit data
                .ENDR
PIBUF:          .BLKB   XMTBUFLEN
;
;  This is the start of the program section
;
START:: .WORD   ^
        $OPEN   FAB=DMOFAB                      ; Open output
        BLBC    R0,EXIT                         ;
        $CONNECT FAB=DMOFAB                     ; Connect to output
        BLBC    R0,EXIT                         ;
        BRB     CONT                            ; Continue
EXIT:   $EXIT_S                                 ; Exit program

CONT:   TYPE    <CSVll Test Program>
        TYPE    < >
        MOVZWL  #1,CHNL                         ; Set up unit no.
        $ASSIGN_S DEVNAM=DEVDSC,CHAN=CHNL       ; Assign unit
        BLBC    R0,EXIT                         ; Exit on error
;
;  Initialize and start controller
;
        MOVZWL  #KMSM_CHR_LOOPB,PIBUF-4         ; Set PI flags, loopback
        MOVW    #XMTBUFLEN,PIBUF-2              ; Set PI buffer size
        MOVZWL  #NMASC_PCLI_PRO,PIBUF           ; PI Protocol
        MOVL    #NMASC_LINPR_PCI,PIBUF-6        ; PI DDCMP Point to Point
        BSBW    INIT                            ; Issue QIO
                                                ; Buffer
;
;  Establish and start DDCMP
;
        CLRQ    R1BUF                           ; Reset PI buffer
        BSBW    ESTAB                           ; Issue QIO
```

```
;
;  Loopback data
;
        MOVZWL    #10,R9                          ; Loop device 10
                                                  ; times
10$:    BSBW      XMIT                            ; Issue transmit
        BSBW      RECV                            ; Issue receive
        MOVAB     XMTBUF,R1                       ; Get address of
                                                  ; transmit data
        MOVAB     RCVBUF,R2                       ; Get address of
                                                  ; received data
        MOVZWL    #XMTBUFLEN,R3                   ; Get number of bytes
                                                  ; to verify
20$:    CMPB      (R1)+,(R2)+                     ; Check data
        BNEQ      30$                             ;
        SOBGTR    R3,20$                          ;
        SOBGTR    R9,10$                          ;
        BRW       EXIT                            ; Exit
30$:    TYPE      <*** Loopback buffer comparison error ***>
        BRW       EXIT                            ; Exit
;
;  Initialize controller QIO
;
INIT:   TYPE      <*** Initialize controller QIO ***>
        $QIO_S    func=#io$_setmode!io$m_ctrl!io$m_startup, chan=chnl,-

                  iosb=iosb,p1=p1buf,p2=#p2bufdsc,p3=#20
        BRW       QIO_STATUS                      ;
;
;  Start DDCMP
;
ESTAB:  TYPE      <*** Startup DDCMP QIO ***>
        $QIO_S    chan=chnl,func=#io$_setmode!io$m_startup,-
                  p1=p1buf,iosb=iosb
        BRW       QIO_XMTST
;
;  Transmit data QIO
;
XMIT:   TYPE      <*** Transmit buffer QIO ***>
        $QIOW_S    chan=chnl,func=#io$_writevblk,p1=xmtbuf,-
                  p2=#xmtbuflen,iosb=iosb
        BRW       QIO_STATUS
;
;  Receive data QIO
;
RECV:   TYPE      <*** Receive buffer QIO ***>
        $QIO_S    chan=chnl,efn=#2,func=#io$_readvblk,p1=rcvbuf,-
                  p2=#xmtbuflen,iosb=iosb
```

```
        .ENABL  LSB
QIO_STATUS:
        BLBC    IOSB,10$                ; Check status of QIO
                                        ; Br if error on QIO
QIO_XMTST:
        BLBC    R0,10$                  ; Check status of XMIT
                                        ; Br if error on
        RSB                             ; request, else return
                                        ; to caller

10$:    PUSHL   IOSB                    ; Get I/O status block
        PUSHL   IOSB+4                  ; Push I/O status block
        PUSHL   R0                      ; Push system service
                                        ; status
        PUSHAQ  FACBUFDSC               ; Push address of FAC
                                        ; buffer descriptor
        PUSHAW  FACLEN                  ; Push address of
                                        ; output length
        PUSHAQ  QIOFEQDSC               ; Push address of
                                        ; input string
        CALLS   #5,G^SYS$FAC            ; Get error message
        MOVAB   CMDBUF,CMDCRAB+RAB$L_RBF ; Get output buffer
                                        ; address
        MOVW    FACLEN,CMDCRAB+RAB$W_RSZ ; Get output buffer
                                        ; length
        $PUT    CMDCRAB                 ; Print error text
        BRW     EXIT                    ; Exit
        .DSABL  LSB

        .END    START
```

# Index

## E



## F



## G



## H

## I



## L



## M



## N

DSV11 Synchronous Device
Driver Manual
AA-JG79B-TE

READER'S COMMENTS

What do you think of this manual? Your comments and suggestions will help us to improve the quality and usefulness of our publications.

Please rate this manual:

| | Poor | | | | Excellent |
|---|---|---|---|---|---|
| Accuracy | 1 | 2 | 3 | 4 | 5 |
| Readability | 1 | 2 | 3 | 4 | 5 |
| Examples | 1 | 2 | 3 | 4 | 5 |
| Organization | 1 | 2 | 3 | 4 | 5 |
| Completeness | 1 | 2 | 3 | 4 | 5 |

Did you find errors in this manual? If so, please specify the error(s) and page number(s).

_____

_____

_____

_____

General comments:

_____

_____

_____

_____

Suggestions for improvement:

_____

_____

_____

_____

Name _____ Date _____

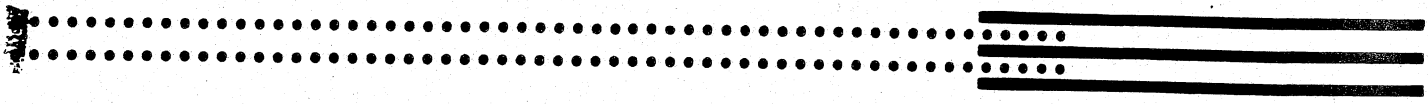
Title _____ Department _____

Company _____ Street _____

City _____ State/Country _____ Zip Code _____

digital™

Digital Equipment Company Limited
Networks and Communications Europe
PO Box 121 READING
Berkshire RG2 0TU
England



digital™

AA-JG798-TE