This book provides reference information for X.25 and X.29 programmers.

Revision/Update Information: This revised manual supersedes X.25 for OpenVMS—Programming Reference, Version 1.1b.

Operating System: OpenVMS I64 Version 8.2 and 8.2-1
OpenVMS Alpha Version 8.2

Software Version: HP X.25 for OpenVMS
Version 2.0
Preface ............................................................ vii

1 Introduction
1.1 System Services ............................................. 1–1
  1.1.1 Format of System Service Descriptions ................. 1–3
  1.1.2 Syntax Conventions ..................................... 1–4
  1.1.3 Common QIO Arguments ................................ 1–4
1.2 Status Codes Returned at System Service Completion ......... 1–5
  1.2.1 Common QIO Return Status Codes ....................... 1–5

2 Common System Services
2.1 $ASSIGN — Assign a Channel ................................. 2–2
2.2 $CANCEL — Clear a Virtual Call on a Channel ............... 2–4
2.3 $CREMBX — Create a Mailbox and Assign a Channel ......... 2–5
2.4 $DASSGN — Deassign the Channel ............................ 2–7
2.5 $GETDVI — Get NV Device Number or Remote DTE Address ... 2–8
  2.5.1 $GETDVI: Get NW or NV Unit Number .................. 2–9
  2.5.2 $GETDVI: Get Remote DTE Address of PAD Connected to NV ... 2–10
2.6 $QIO(IO$_ACCESS) — Set Up a Virtual Circuit ................. 2–11
2.7 $QIO(IO$_ACCESS!IO$_M_ABORT) — Reject a Call ............. 2–19
2.8 $QIO(IO$_ACCESS!IO$_M_ACCEPT) — Accept a Call ............ 2–23
2.9 $QIO(IO$_ACCESS!IO$_M_REDIRECT) — Redirect a Call ...... 2–28
2.10 $QIO(IO$_ACPCONTROL) — Declare a Network Process ....... 2–31
2.11 $QIO(IO$_DEACCESS) — Clear a Virtual Circuit .......... 2–39
2.12 $QIO(IO$_READVBLK) — Receive Data ...................... 2–43
2.13 $QIO(IO$_WRITEVBLK) — Transmit Data .................... 2–45

3 X.25 System Services
3.1 $QIO(IO$_NETCONTROL, PSI$K_INTACK) — Confirm Receipt of an ... 3–2
  Interrupt ..................................................... 3–2
3.2 $QIO(IO$_NETCONTROL, PSI$K_INTERRUPT) — Transmit an ... 3–4
  Interrupt .................................................... 3–4
3.3 $QIO(IO$_NETCONTROL, PSI$K_RESET) — Reset a Virtual Circuit or ... 3–6
  Confirm the Receipt of a Reset .............................. 3–6
3.4 $QIO(IO$_NETCONTROL, PSI$K_RESTART) — Confirm Receipt of a ... 3–8
  Restart ....................................................... 3–8
4 X.29 System Services

4.1 \$QIO(IO$_\text{NETCONTROL}, PSI$K_X29$\_READ) — Read X.29 Terminal Characteristics ............................................. 4–3

4.1.1 READ Subfunction: PSI$K_X29$\_BREAK$\_ACTION ......................... 4–4
4.1.2 READ Subfunction: PSI$K_X29$\_HANGUP$\_PARAMS .................... 4–5
4.1.3 READ Subfunction: PSI$K_X29$\_HOLD$\_TIMER ......................... 4–6
4.1.4 READ Subfunction: PSI$K_X29$\_HOST$\_ECHO$\_PARAMS ............... 4–7
4.1.5 READ Subfunction: PSI$K_X29$\_INT$\_ACTION ......................... 4–8
4.1.6 READ Subfunction: PSI$K_X29$\_LOCAL$\_ECHO$\_PARAMS .............. 4–9
4.1.7 READ Subfunction: PSI$K_X29$\_PAD$\_PARAMS .................... 4–10
4.1.8 READ Subfunction: PSI$K_X29$\_TEMP$\_NOHANG ............... 4–12

4.2 \$QIO(IO$_\text{NETCONTROL}, PSI$K_X29$\_READ$\_SPECIFIC) — Read Specific X.29 Parameters ............................................. 4–13

4.2.1 READ$\_SPECIFIC$ Subfunction: PSI$K_X29$\_PAD$\_PARAMS ........... 4–14

4.3 \$QIO(IO$_\text{NETCONTROL}, PSI$K_X29$\_SET) — Set X.29 Terminal Characteristics .................................................. 4–16

4.3.1 SET Subfunction: PSI$K_X29$\_BREAK$\_ACTION ......................... 4–18
4.3.2 SET Subfunction: PSI$K_X29$\_HANGUP$\_PARAMS .................... 4–20
4.3.3 SET Subfunction: PSI$K_X29$\_HOST$\_ECHO$\_PARAMS ............... 4–21
4.3.4 SET Subfunction: PSI$K_X29$\_HOLD$\_TIMER ......................... 4–23
4.3.5 SET Subfunction: PSI$K_X29$\_INT$\_ACTION ......................... 4–24
4.3.6 SET Subfunction: PSI$K_X29$\_LOCAL$\_ECHO$\_PARAMS ............... 4–25
4.3.7 SET Subfunction: PSI$K_X29$\_PAD$\_PARAMS .................... 4–27
4.3.8 SET Subfunction: PSI$K_X29$\_PAD$\_RESELECTION ................. 4–29
4.3.9 SET Subfunction: PSI$K_X29$\_TEMP$\_NOHANG ............... 4–31

5 Status Codes Returned at System Service Completion

5.1 Testing the Return Status Code ............................................. 5–3
5.2 Special Return Conditions ............................................... 5–4
5.2.1 Resource Wait Mode .................................................. 5–4
5.2.2 System Service Failure Exception Mode ......................... 5–4
5.3 Obtaining Values for Other Symbolic Codes ....................... 5–5

A Summary of X.25 System Service Calls

A.1 System Services for Setting Up and Clearing Communications .............. A–1
A.2 System Services for Handling Incoming Calls .......................... A–1
A.3 System Services for Handling Control and Data Messages ............... A–2

B Summary of X.29 System Service Calls

B.1 System Services for Setting Up and Clearing Communication .............. B–1
B.2 System Services for Handling Incoming Calls .......................... B–1
B.3 System Services for Reading and Setting PAD Parameters and NV Terminal Characteristics ....................... B–2
B.4 Terminal Driver Functions ............................................. B–2
Preface

Manual Objectives

This manual provides reference information for X.25 and X.29 programmers.

The information in this manual applies to the X.25 functionality provided by HP X.25 for OpenVMS and HP DECnet–Plus for OpenVMS VAX. Note that the X.25 functionality in DECnet–Plus for OpenVMS VAX was formerly provided by VAX P.S.I. software.

Throughout this manual, the X.25 functionality provided by both HP X.25 for OpenVMS and HP DECnet–Plus for OpenVMS VAX is referred to generically as X.25 for OpenVMS.

Audience

This manual is intended for programmers who perform network operations. The manual assumes that you have knowledge and experience of the following:

- OpenVMS operating system
- OpenVMS system services
- Packet switching
- DECnet–Plus
- A programming language

The manual also assumes that you have some knowledge of general communications theory, and that you understand X.25 and PSDN terminology.

Structure

The manual is divided into five chapters and six appendices:

- Chapter 1 introduces the system services used for X.25 and X.29 programming, and explains the structure of the reference information.
- Chapter 2 details the system services common to both X.25 and X.29 programming.
- Chapter 3 details the system services specific to X.25 programming.
- Chapter 4 details the system services specific to X.29 programming.
- Chapter 5 describes the status values returned by the system services used for X.25 and X.29 programming.
- Appendix A provides a summary of the format of the X.25 system services.
- Appendix B provides a summary of the format of the X.29 system services.
- Appendix C describes the structure of the Network Connect Block.
• Appendix D provides reference information about mailbox messages.
• Appendix E contains descriptions of the standard PAD parameters.
• Appendix F describes the X.25 and X.29 programming examples that are provided in `SYS$EXAMPLES`.

Associated Manuals

The following sections describe HP DECnet–Plus for OpenVMS, HP X.25 for OpenVMS, and HP OpenVMS manuals that either directly describe the X.25 for OpenVMS software or provide related information.

HP DECnet–Plus for OpenVMS Documentation

The following DECnet–Plus manuals contain information useful to X.25 for OpenVMS managers, users, and programmers:

• **HP DECnet–Plus for OpenVMS —Introduction and User’s Guide**
  This manual provides general information on DECnet–Plus and describes the concept of packet switching data networks.

• **HP DECnet–Plus for OpenVMS—Installation and Configuration**
  This manual describes how to install and configure DECnet–Plus for OpenVMS software. For OpenVMS I64 and OpenVMS Alpha systems, this manual also describes how to install X.25 for OpenVMS software. Details on configuring X.25 for OpenVMS on OpenVMS I64 and OpenVMS Alpha systems are provided in the **HP X.25 for OpenVMS—Configuration** manual. For OpenVMS VAX systems, this manual also describes how to install and configure the X.25 functionality provided by DECnet–Plus for OpenVMS VAX.

• **HP DECnet–Plus for OpenVMS—Network Management**
  This manual provides conceptual and task information about managing and monitoring a DECnet–Plus network. In addition, the manual devotes a section to the management of X.25 entities used by DECnet operating over X.25 data links.

• **HP DECnet–Plus for OpenVMS—Network Control Language Reference**
  This manual provides detailed information on the Network Control Language (NCL), which is used to manage X.25 for OpenVMS management entities.

HP X.25 for OpenVMS Documentation

The following manuals make up the X.25 for OpenVMS documentation set:

• **HP X.25 for OpenVMS—Configuration** (OpenVMS I64 and OpenVMS Alpha)
  This manual explains how to configure X.25 for OpenVMS software on OpenVMS I64 and OpenVMS Alpha systems.

• **HP X.25 for OpenVMS—Security Guide**
  This manual describes the X.25 Security model and how to set up, manage, and monitor X.25 Security to protect your X.25 for OpenVMS system from unauthorized incoming and outgoing calls.

• **HP X.25 for OpenVMS—Problem Solving Guide**
  This manual provides guidance on how to analyze and correct X.25–related and X.29–related problems that may occur while using the X.25 for OpenVMS software. In addition, the manual describes loopback testing for LAPB data links.
HP X.25 for OpenVMS—Programming Guide
This manual describes how to write X.25 and X.29 programs to perform network operations.

HP X.25 for OpenVMS—Programming Reference
This manual provides reference information for X.25 and X.29 programmers. It is a companion manual to the HP X.25 for OpenVMS—Programming Guide.

HP X.25 for OpenVMS—Utilities Guide
This manual describes how to use and manage X.25 Mail and how to use and manage a host–based PAD to connect to a remote system. It also describes how to manage the X.29 communication links used for both of these functions. In addition, this manual explains how to use OpenVMS DCL SET TERMINAL/X29 commands to manage remote host–based or network PADs.

HP X.25 for OpenVMS—Accounting
This manual describes how to use X.25 Accounting to obtain performance records and information on how X.25 is being used on your system.

HP OpenVMS Documentation
The following OpenVMS manuals contain information useful to X.25 for OpenVMS managers, users, and programmers:

• The current HP OpenVMS New Features and Documentation Overview manual
• HP OpenVMS DCL User’s Manual
• HP OpenVMS DCL Dictionary
• HP OpenVMS System Management Utilities Reference Manual
• HP OpenVMS System Services Reference Manual
• HP OpenVMS Guide to System Security

Reader’s Comments
HP welcomes your comments on this manual or any of the X.25 for OpenVMS documents. Please send comments to either of the following addresses:

Internet  openvmsdoc@hp.com
Mail  Hewlett-Packard Company
      OSSG Documentation Group, ZKO3-4/U08
      110 Spit Brook Rd.
      Nashua, NH 03062-2698

How To Order Additional Documentation
For information about how to order additional documentation and for online versions of most X.25 for OpenVMS documentation, visit the following World Wide Web address:

http://www.hp.com/go/openvms/doc/
**Terminology**

The terminology used in the VAX P.S.I. product has been replaced by the terminology used in the X.25 for OpenVMS product. Table 1 shows the correlation between VAX P.S.I. terms and their X.25 for OpenVMS counterparts.

<table>
<thead>
<tr>
<th>VAX P.S.I.</th>
<th>X.25 for OpenVMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAX P.S.I.</td>
<td>X.25 for OpenVMS VAX</td>
</tr>
<tr>
<td>Access system</td>
<td>X.25 Client system</td>
</tr>
<tr>
<td>Native system</td>
<td>X.25 Direct Connect system</td>
</tr>
<tr>
<td>Multihost system</td>
<td>X.25 Connector system</td>
</tr>
<tr>
<td>Gateway system</td>
<td>X.25 Connector system</td>
</tr>
</tbody>
</table>

In addition to the terms shown in Table 1, the X.25 for OpenVMS documentation set uses the following standard terms for client systems, server systems, relay systems, and the X.25 for OpenVMS management entities that represent these systems:

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client system</td>
<td>A client system of an X.25 Connector system (and therefore a client of the X25 Server management module on the X.25 Connector system.)</td>
</tr>
<tr>
<td>Relay Client system</td>
<td>A client system of an X.25 Relay system (and therefore a client of the X25 Relay management module on the X.25 Relay system.)</td>
</tr>
<tr>
<td>Relay–Client</td>
<td>A shorthand term for an X25 RELAY CLIENT management entity on an X.25 Relay system that contains management information about an actual Relay Client system.</td>
</tr>
<tr>
<td>Relay system</td>
<td>An X.25 Direct Connect or Connector system with the X.25 Relay module enabled.</td>
</tr>
<tr>
<td>Server Client system</td>
<td>Another term for a Client system.</td>
</tr>
<tr>
<td>Server–Client</td>
<td>A shorthand term for an X25 SERVER CLIENT management entity on an X.25 Connector system that contains management information about one or more actual X.25 Client systems.</td>
</tr>
</tbody>
</table>

For more information about clients, servers, and relays in X.25 for OpenVMS, refer to the *HP X.25 for OpenVMS—Configuration* manual and the *HP X.25 for OpenVMS—Management Guide*. 
Conventions

The following conventions are used in the X.25 for OpenVMS documentation set:

**UPPERCASE and lowercase**

The OpenVMS operating system does not differentiate between lowercase and uppercase characters. Literal strings that appear in text, examples, syntax descriptions, and function descriptions can be entered using uppercase characters, lowercase characters, or a combination of both.

In running text, uppercase characters indicate OpenVMS DCL commands and command qualifiers; Network Control Language (NCL) commands and command parameters; other product-specific commands and command parameters; network management entities; OpenVMS system logical names; and OpenVMS system service calls, parameters, and item codes.

Leading uppercase characters, such as Protocol State, indicate management entity characteristics and management entity event names. Leading uppercase characters are also used for the top-level management entities known as modules.

**system output**

This typeface is used in interactive and code examples to indicate system output. In running text, this typeface is used to indicate the exact name of a device, directory, or file; the name of an instance of a network management entity; or an example value assigned to a DCL qualifier or NCL command parameter.

**user input**

In interactive examples, user input is shown in **bold** print.

$  

In this manual, a dollar sign ($) is used to represent the default OpenVMS user prompt.

Ctrl/X  

In procedures, a sequence such as Ctrl/X indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.

Return  

In procedures, a key name is shown enclosed to indicate that you press the corresponding key on the keyboard.

**italic text**

Italic text indicates variables or book names. Variables include information that varies in system input and output. In discussions of event messages, italic text indicates a possible value of an event argument.

**bold text**

Bold text indicates an important term, or important information.

( )  

In a command definition, parenthesis indicate that you must enclose the options in parenthesis if you choose more than one. Separate the options using commas.

{ }  

In a command definition, braces are used to enclose sets of values. The braces are a required part of the command syntax.

[ ]  

In a command definition, square brackets are used to enclose parts of the command that are optional. You can choose one, none, or all of the options. The brackets are not part of the command syntax. However, brackets are a required syntax element when specifying a directory name in an OpenVMS file specification.

**Note**

The following conventions apply to multiplatform documentation.
Indicates information specific to OpenVMS I64 and OpenVMS Alpha. Note that single lines of information specific to OpenVMS I64 and OpenVMS Alpha are marked “(OpenVMS I64 and OpenVMS Alpha)” or “(OpenVMS I64/Alpha)”.

Indicates information specific to OpenVMS VAX. Note that single lines of information specific to OpenVMS VAX are marked “(OpenVMS VAX)”.

 Indicates the end of platform–specific information.
X.25 for OpenVMS provides a set of system services for you to communicate with a remote DTE.

This chapter introduces you to the available system services, the format of the system service descriptions in this manual, and the syntax conventions used.

### 1.1 System Services

Table 1–1 summarizes the system services that can be used for X.25 and X.29 programming. Details of each system service are given in the following sections:

- Chapter 2 details the system services common to both X.25 and X.29 programming.
- Chapter 3 details the system services specific to X.25 programming.
- Chapter 4 details the system services specific to X.29 programming.

Note

All constants in the program header files associated with the system services are defined in lowercase.

Example X.25 and X.29 programs are provided in the $SYS$EXAMPLES: directory. Appendix F describes the available example programs.
Table 1–1 System Services

<table>
<thead>
<tr>
<th>System Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Services Common to X.25 and X.29 Programming</strong> (Refer to Chapter 2)</td>
<td></td>
</tr>
<tr>
<td>$ASSIGN</td>
<td>Assigns a Channel</td>
</tr>
<tr>
<td>$CANCEL</td>
<td>Clears a Virtual Call on a Channel</td>
</tr>
<tr>
<td>$CREMBX</td>
<td>Creates a Mailbox and Assigns a Channel</td>
</tr>
<tr>
<td>$DASSGN</td>
<td>Deassigns the Channel</td>
</tr>
<tr>
<td>$GETDVI</td>
<td>Gets the NV Device Number or Remote DTE Address</td>
</tr>
<tr>
<td>$QIO(IO$_ACCESS)</td>
<td>Sets Up a Virtual Circuit</td>
</tr>
<tr>
<td>$QIO(IO$_ACCESS!IO$M_ABORT)</td>
<td>Rejects a Call</td>
</tr>
<tr>
<td>$QIO(IO$_ACCESS!IO$M_ACCEPT)</td>
<td>Accepts a Call</td>
</tr>
<tr>
<td>$QIO(IO$_ACCESS!IO$M_REDIRECT)</td>
<td>Redirects a Call</td>
</tr>
<tr>
<td>$QIO(IO$_ACPCONTROL)</td>
<td>Declares a Network Process</td>
</tr>
<tr>
<td>$QIO(IO$_DEACCESS)</td>
<td>Clears a Virtual Circuit</td>
</tr>
<tr>
<td>$QIO(IO$_READVBLK)</td>
<td>Receives Data</td>
</tr>
<tr>
<td>$QIO(IO$_WRITEVBLK)</td>
<td>Transmits Data</td>
</tr>
<tr>
<td><strong>Services Specific to X.25 Programming</strong> (Refer to Chapter 3)</td>
<td></td>
</tr>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_INTACK)</td>
<td>Confirms Receipt of an Interrupt</td>
</tr>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_INTERRUPT)</td>
<td>Transmits an Interrupt</td>
</tr>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_RESET)</td>
<td>Resets a Virtual Circuit or Confirms Receipt of a Reset</td>
</tr>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_RESTART)</td>
<td>Confirms Receipt of a Restart</td>
</tr>
<tr>
<td><strong>Services Specific to X.29 Programming</strong> (Refer to Chapter 4)</td>
<td></td>
</tr>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_X29_READ)</td>
<td>Reads X.29 Terminal Characteristics</td>
</tr>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_X29_READ_SPECIFIC)</td>
<td>Reads Specific X.29 Parameters</td>
</tr>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_X29_SET)</td>
<td>Sets X.29 Terminal Characteristics</td>
</tr>
</tbody>
</table>
1.1.1 Format of System Service Descriptions

In Chapters 2, 3, and 4, the system services are arranged in alphabetical order. Each system service description contains an outline of the function of the service, plus the following items, where applicable:

**Format**
Shows the macro name, with all keyword arguments listed in order of position.

**Arguments**
Describes the arguments. Arguments which are unique to X.25 or X.29 operation are described for each system service. Arguments which are common for all calls are described in Section 1.1.3.

**NCB Contents**
Lists the mandatory, optional, and ignored items contained in the Network Connect Block (NCB). Ignored items are those ignored by the X.25 for OpenVMS software. If you include other items in the NCB, an error is reported.

**Examples**
Shows the system service with arguments completed in MACRO–style code. These examples are very general and you are recommended to refer to your programming language manual for specific details of implementing the system service. Example programs are provided in the SYS$EXAMPLES: directory. Refer to Appendix F.

**Return Status**
Lists those status codes returned by the service that apply to X.25 for OpenVMS, and explains what the return status codes mean. Common status codes that may be returned are listed in Section 1.2.1.

______________________________ Note ________________________________

In Chapters 2, 3, and 4, notes that are referred to in the text and tables are presented at the end of the system service to which they refer.
1.1.2 Syntax Conventions

The following conventions are used in this manual to describe the syntax of the system services.

1. A character is one of the set of alphanumerics that includes:
   - A to Z
   - a to z
   - 0 to 9
   - _ (underscore)
   - $ (dollar)

2. All system service names are in UPPERCASE letters, and you must enter these as shown. Arguments are in italics, and you must replace the argument shown in the system service format with the precise information requested.

3. Square brackets [ ] enclose optional keywords and arguments. Do not include the brackets when entering the system service.

4. You must enter punctuation such as commas and parentheses ( ) as shown in the format. Use consecutive commas to indicate omitted arguments; you can omit commas indicating optional arguments at the end of a system service macro.

1.1.3 Common QIO Arguments

Only those arguments which are unique to X.25 or X.29 operations are described for each system service. The arguments listed below are common to all system services, and always have the values shown here:

- **efn**
  
  Number of the event flag to be set at request completion. If not specified, the default is 0.
  
  When QIO begins execution, it clears the specified event flag. The event flag is set even if the service terminates without queuing an I/O request.

- **chan**
  
  I/O channel that is assigned to the device to which the request is directed. The chan argument is a word value containing the number of the I/O channel.

- **iosb**
  
  I/O status block to receive the final completion status of the I/O operation. iosb is the address of the quadword I/O status block (see below).

- **astadr**
  
  AST service routine to be executed when the I/O completes. The astadr argument is the address of a longword value that is the entry mask to the AST routine. The AST routine executes at the access mode of the caller of QIO.

- **astprm**
  
  AST parameter to be passed to the AST service routine. The astprm argument is a longword value containing the AST parameter.

For example, for a QIO with the following format, the Arguments section of the system service description describes only the argument **func**, and parameters p1 to p6:

```
$QIO efn,chan,func,iosb,astadr,astprm,p1, p2, p3, p4, p5, p6
```

The other arguments have the values given in the preceding table.
1.2 Status Codes Returned at System Service Completion

When you request a system service, the status returned to Register 0 (R0) indicates only whether the request was queued successfully. To check whether a system service has completed successfully, your program should also examine the first word of the I/O Status Block (IOSB). There may be further status information in words 2 and 3. Refer to Table 5–1.

For further information about return status codes, refer to Chapter 5.

1.2.1 Common QIO Return Status Codes

Only those return status codes that are unique to X.25 or X.29 operations are described for each system service. Unless they are described as having a meaning specific to X.25 for OpenVMS, status codes have the meanings given below. The status codes are listed in alphabetical order.

Three of the error codes are **severe**. They indicate an immediate failure, because OpenVMS cannot process your system service call. These errors are indicated by the following status codes:

- **SS$$_ACCVIO** Either of the following:
  - The argument list, device, mailbox name string, string descriptor, buffer, or IOSB cannot be read by the caller.
  - The channel number, buffer, or IOSB cannot be written by the caller. If the argument list cannot be read by the caller (using the $name_G form), the service is not called. This is a particular meaning of SS$$_ACCVIO. It is different from the meaning listed for many individual system services, in which the service is called, but one or more specific arguments are addresses that cannot be read or written by the caller.

- SS$_ILLSER An illegal system service was called.

- SS$_INSFARG Not enough arguments were supplied to the service.

In addition to the above status codes, and the codes that apply specifically to X.25 for OpenVMS described with each service, the services may return one or more of the following codes:

- **SS$$_ABORT** PSDN, DECnet or X.25 for OpenVMS software failed during request processing.

- **SS$$_BADPARAM** One or more of the parameters $p1 to $p6 is not valid for this QIO.

- **SS$$_CANCEL** A $CANCEL service was issued for this channel while the request was being processed.

- **SS$$_EXQUOTA** The process does not have sufficient buffered I/O quota or other resources to complete the request.

- **SS$$_ILLEFC** An illegal event flag number was specified.

- **SS$$_ILLIOFUNC** The function code was illegal, or had illegal or conflicting modifier bits set.

- **SS$$_INSFMEM** There is insufficient system dynamic memory to complete the request.

- SS$$_IVBUFLEN The format of the NCB item–list is invalid.

- SS$$_IVCHAN An invalid channel number was specified.
SS$_NOPRIV  The process does not have the privileges required for this function.

SS$_NORMAL  Service completed successfully.

SS$_UNASEFC The process is not associated with the cluster containing the specified event flag.

For more information on return status codes, refer to the OpenVMS system services documentation.
Table 2–1 summarizes the system services common to both X.25 and X.29 programming.

<table>
<thead>
<tr>
<th>System Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ASSIGN</td>
<td>Assigns a Channel</td>
</tr>
<tr>
<td>$CANCEL</td>
<td>Clears a Virtual Call on a Channel</td>
</tr>
<tr>
<td>$CREMBX</td>
<td>Creates a Mailbox and Assigns a Channel</td>
</tr>
<tr>
<td>$DASSGN</td>
<td>Deassigns the Channel</td>
</tr>
<tr>
<td>$GETDVI</td>
<td>Gets the NV Device Number or Remote DTE Address</td>
</tr>
<tr>
<td>$QIO(IO)_ACCESS)</td>
<td>Sets Up a Virtual Circuit</td>
</tr>
<tr>
<td>$QIO(IO)_ACCESS!IO$M_ABORT)</td>
<td>Rejects a Call</td>
</tr>
<tr>
<td>$QIO(IO)_ACCESS!IO$M_ACCEPT)</td>
<td>Accepts a Call</td>
</tr>
<tr>
<td>$QIO(IO)_ACCESS!IO$M_REDIRECT)</td>
<td>Redirects a Call</td>
</tr>
<tr>
<td>$QIO(IO)_ACPCONTROL)</td>
<td>Declares a Network Process</td>
</tr>
<tr>
<td>$QIO(IO)_DEACCESS)</td>
<td>Clears a Virtual Circuit</td>
</tr>
<tr>
<td>$QIO(IO)_READVBLK)</td>
<td>Receives Data</td>
</tr>
<tr>
<td>$QIO(IO)_WRITEVBLK)</td>
<td>Transmits Data</td>
</tr>
</tbody>
</table>
2.1 $ASSIGN — Assign a Channel

$ASSIGN obtains a channel number, and associates a (previously created) mailbox with the channel.

In X.25 programs, you use $ASSIGN to assign a channel to the NW device.

In an X.29 program, you use $ASSIGN to assign a channel to the NW device or to the NV device.

When your program attempts to assign a channel to NWA0: or NVA0: X.25 for OpenVMS creates a new device called NWA\textsubscript{uu} or NVA\textsubscript{uu}: (where \textit{uu} is a unique unit number), and assigns the channel to that device. $ASSIGN never assigns a channel to NWA0: or NVA0:.

You will use the number allocated to the NW device in all QIOs which communicate with a remote DTE. In X.29 programs, you will need to supply the number of the NV device as the \textit{p6} parameter in all QIO calls to the NW device. Use $GETDVI to discover the unit number allocated to the NV or NW device.

Note that your program must assign \textbf{only one channel} for each virtual circuit to the NWA\textsubscript{uu} device.

In X.25 programs, for a Permanent Virtual Circuit (PVC) you assign a channel to the device NWA0: (exactly as for an SVC), and then specify the name of the PVC in the NCB for the $QIO(IO\_ACCESS) service.

\textbf{Format}

$ASSIGN \textit{devnam},\textit{chan},[\textit{acmode}],[\textit{mbxnam}]

\textbf{Arguments}

\textit{devnam} Address of a quadword character string descriptor pointing to the device name string.

For X.25 programs, the character string contains NWA0: or a logical name for NWA0:.

For X.29 programs, the character string contains either NWA0: or NVA0:, or a logical name definition for either.

\textit{chan} Address of a word to receive the channel number assigned.

\textit{acmode} Access mode to be associated with the channel. The specified access mode must be an access mode less privileged than, or equal in privilege to, the access mode from which the service was called. The channel allows I/O operations only from equally privileged, or more privileged, access modes.

\textit{mbxnam} Address of a quadword character string descriptor pointing to the logical name string for the mailbox, if required, to be associated with the channel. An address of 0 implies no mailbox; this is the default value. This mailbox remains associated with the channel until you delete the mailbox or deassign the channel ($DASSGN)$.
Example
In the following example, the device name is referred to by PSI_DEV. The channel
to the network device is placed in PSI_CHAN. A mailbox, MBX, is assigned to this
channel. No access mode is specified.

;Declaring the data:
PSI_DEV:
  .ASCID /_NWA0:/ ; Network device name
PSI_CHAN:
  .BLKW 1 ; Mailbox channel
MBX:
  .ASCID /SYS$NET/ ; Mailbox logical name

; Using the System Service:
$ASSIGN_S - ; Assign a channel
  DEVNAM = PSI_DEV,- ; to network device
  CHAN = PSI_CHAN,- ; Channel number
  MBXNAM = MBX ; Mailbox name

Return Status
SS$_NORMAL Service successfully completed, a channel has been assigned.
SS$_IVDEVNAM Either no device name was specified, or either the device name
or the mailbox name string contains invalid characters.
SS$_IVLOGNAM The device name or mailbox name string has a length of 0 or
has more than 63 characters.
SS$_NOIOCHAN No channel is available for assignment.
SS$_NOSUCHDEV The specified device or mailbox does not exist.
2.2 **$CANCEL — Clear a Virtual Call on a Channel**

The $CANCEL system service cancels all pending I/O requests on the specified channel. This has the effect of clearing the virtual call in progress on the channel.

You can cancel I/O requests only from an access mode equal to, or more privileged than, the access mode from which you originally assigned the channel.

When a request currently in progress is cancelled, the driver is notified immediately. The action taken for I/O requests in progress is similar to that taken for queued requests:

- The specified event flag is set.
- The first word of the IOSB, if specified, is set to SS$_CANCEL if the I/O request is queued or to SS$_ABORT if the I/O request is in progress.
- The AST, if specified, is queued.

Outstanding I/O requests are automatically cancelled at image exit.

**Format**

```
$CANCEL chan
```

**Arguments**

- **chan** Number of the I/O channel on which I/O is to be cancelled.

**Example**

In this example, the call cancels all pending I/O requests to the network device on channel PSI_CHAN.

```plaintext
; Declaring the data:
PSI_CHAN:
   .BLKW 1    ; Mailbox channel

; Using the System Service
$CANCEL_S -         ; Cancel I/O requests
   CHAN = PSI_CHAN   ; on channel
```

**Return Status**

- **SS$_NORMAL** Service successfully completed, I/O has been cancelled on the specified I/O channel.
- **SS$_EXQUOTA** Process does not have sufficient buffered I/O quota and has disabled resource wait mode.
- **SS$_NOPRIV** The specified channel is not assigned, or was assigned from a more privileged access mode.
2.3 $CREMBX — Create a Mailbox and Assign a Channel

$CREMBX creates a virtual mailbox device named MBA\(uu\):, and assigns an I/O channel to it.

The system provides the unit number, \(uu\), when it creates the mailbox MBA\(uu\):. If a mailbox with the specified name already exists, $CREMBX assigns a channel to the that mailbox. It should not however be used to create a channel to the mailbox SYS$NET as $CREMBX does not recognize SYS$NET as an existing mailbox.

Format

$CREMBX [prmflg],chan,[maxmsg],[bufquo],[promsk],[acmode], [lognam]

Arguments

- **prmflg** Specifies whether the mailbox is to be permanent or temporary. This argument is a longword value: 1 for permanent; 0 for temporary.
- **chan** Address of a word to receive the channel number assigned.
- **maxmsg** Maximum size (in bytes) of a message that can be sent to the mailbox. If not specified, or specified as 0, OpenVMS provides a default value.
- **bufquo** Number of bytes of system dynamic memory that can be used to buffer messages sent to the mailbox. This argument is a longword value. For a temporary mailbox, the value must be less than or equal to the buffer quota of the process. OpenVMS provides a default value.
- **promsk** Protection mask, specified by a longword value. Cleared bits grant access to four types of user: world, group, owner, and system. If promsk is not specified, access is granted to all users.
- **acmode** A longword containing one of the four access modes defined by the $PSIDEF macro.
- **lognam** The address of a character string descriptor pointing to a logical name string to be assigned to the mailbox.

Example

In this example $CREMBX is used to create a network device mailbox and assign the network channel MBX_CHAN.

MBX_CHAN:
       .BLKW 1 ; Channel to mailbox
MBX_NAME:
       .ASCID /X29_MBX/ ; Mailbox name

; Create network device mailbox and assign network channel.

$CREMBX_S - ; Create mailbox
   CHAN = MBX_CHAN,- ; channel
   LOGNAM = MBX_NAME ; logical name
   BSBW ERROR ; Check for error
Return Status

SS$_NORMAL  Service completed successfully.
SS$_BADPARAM The *bufquo* argument specified too large a value.
SS$_IVSTSFLG Undefined bit set in the *prmflg* argument. This argument must have a value of 1 or 0.
SS$_NOIOCHAN No channel is available for assignment.
SS$_NOPRIV The process does not have the privilege to create this type of mailbox.
2.4 $DASSGN — Deassign the Channel

$DASSGN deassigns the logical channel to an NV or NW device.

For channels assigned to NV devices, $DASSGN takes the following action:

- If the channel is the only one assigned to NVAuu:, the terminal characteristic /HANGUP is set and PSI$K_X29_TEMP_NOHANG is not set, the channel is released, and the circuit is cleared.

- If the channel is the last one assigned to NVAuu:, the terminal characteristic /TYPEAHEAD is set and PSI$K_X29_TEMP_NOHANG is set, OpenVMS begins the login sequence at the X.29 terminal.

Format

$DASSGN chan

Arguments

chan Number of the channel to be deassigned.

Example

In the following example, channel PSI_CHAN is deassigned.

; Declaring the data:
PSI_CHAN:
   .BLKW 1 ; Mailbox channel

; Using the System Service:
$DASSGN_S - ; Deassign channel
   CHAN = PSI_CHAN ; to network device

Return Status

SS$_NORMAL Service completed successfully, channel has been deassigned.

SS$_NOPRIV The specified channel is not assigned, or was assigned from a more privileged access mode.
2.5 $GETDVI — Get NV Device Number or Remote DTE Address

In X.25 programs, you can use $GETDVI to obtain the NW unit number (see Section 2.5.1).

In X.29 programs, you can use $GETDVI to:

- Obtain the NW unit number (see Section 2.5.1).
- Obtain the NV unit number (see Section 2.5.1).
- Obtain the remote DTE address (see Section 2.5.2).
2.5.1 $GETDVI: Get NW or NV Unit Number

$GETDVI obtains the unit number allocated by the NW or NV device driver.

The NV unit number is important in X.29 programs, because you need to supply the NV unit number as the p6 parameter in any QIO request to NWA0: (the X.25 network device).

For details of how to connect an NV device to a VT device, see the *HP X.25 for OpenVMS—Programming Guide*.

**Format**

$GETDVI [efn],[chan],[devnam],itmlst,[iosb],[astadr], [astprm],nullarg

**Arguments**

*chan* Number of the channel assigned to the NW or NV device.

*itmlst* Address of a descriptor block returning the unit number of the NW or NV device. Specify DVI$$_UNIT$$ as the item code.

**Example**

In the following example, $GETDVI gets the unit number of the NV device, and returns it in the item–list UNIT_LIST.

```plaintext
NV_CHAN:
  .BLKW 1 ; Channel to NV
UNIT_LIST:
  .WORD 4 ; length (in bytes) of buffer
  .WORD DVI$$_UNIT$$ ; item code
  .LONG NV_UNIT ; Address of buffer for $GETDVI
  .LONG NV_UNIT_LENGTH ; Address of word for $GETDVI
  .LONG 0 ; to put the amount of data output
IO_STATUS:
  .BLKW 4 ;I/O Status block

;+ ; Get the unit number of the NV device.
;-
$GETDVIW_S - ; Use this routine to convert
  ; the channel number given to
  ; a unit number.
  CHAN = NV_CHAN,- ; channel
  ITMLST = UNIT_LIST ; Address of item-list of information.
  IOSB = IO_STATUS ; Status return
  BSBW ERROR ; Check for error
```

**Return Status**

SS$_$NORMAL Service completed successfully.
2.5.2 $GETDVI: Get Remote DTE Address of PAD Connected to NV

In X.29 programs, you can use $GETDVI to return a string containing the remote DTE address of the calling PAD and the name of the local DTE class that the call was received on.

The remote DTE will only be returned if the PSDN provided the remote DTE address in the CALL packet. The local DTE class is always returned. The format of the returned string is:

dte-class.remote-dte-address

Format

$GETDVI [efn],[chan],[devnam],itmlst,[iosb],[astadr], [astprm],nullarg

Arguments

- **chan**: Number of the channel assigned to the NV device.
- **itmlst**: Address of a descriptor block returning the local DTE class and the remote DTE address of the PAD for the NV device. Specify DVI$_TT_ACCPORNAM as the item code.

Example

In this example, $GETDVI gets the remote DTE address of the PAD, and returns it in the item–list REMDTE_LIST.

```
PAD_REMDTE_LENGTH: .blkw 1 ; Length of returned string
PAD_REMDTE: .BLKB 64 ; Storage To hold the remote DTE address in
NV_CHAN:
  .BLKW 1 ; Channel to NV
REMDTE_LIST:
  .WORD 64 ; length (in bytes) of buffer
  .LONG DVI$_TT_ACCPORNAM ; item code
  .LONG PAD_REMDTE ; Address of buffer for $GETDVI
  .LONG PAD_REMDTE_LENGTH ; Address of word for $GETDVI
  .LONG 0 ; to put the amount of data output
IO_STATUS:
  .BLKW 4 ; I/O Status block

;+ ; Get the remote DTE address of the PAD
;-
$GETDVIW_S - ; Use this routine to obtain
  ; the local DTE class and
  ; remote DTE.
  CHAN = NV_CHAN,-
  ITMLST = REMDTE_LIST ; Address of item-list of information.
  -
  IOSB = IO_STATUS ; Status return
  BSBW ERROR ; Check for error

Return Status

SS$_NORMAL Service completed successfully.
2.6 $QIO(IO$_ACCESS) — Set Up a Virtual Circuit

The QIO system service with a function code of IO$_ACCESS requests a virtual circuit to be set up, and can optionally request network facilities. If you subscribe to the fast select facility, up to 128 bytes of data can be sent with the request rather than being limited to 16 bytes of data for normal calls. You must also use this service before data can be transmitted or received on a PVC.

For an SVC, the service completes when the request is either accepted or rejected by the remote DTE. For a PVC, the call completes without any PSDN activity. If there is a mailbox associated with the NW device, an NCB is written to the mailbox with details of the call acceptance or call rejection. Call accept messages have the code MSG$_CONNECT, and reject messages have the code MSG$_DISCONNECT.

If the rights identifier PSI$X25_USER is defined on your system, your program must possess either that rights identifier or BYPASS privilege.

If PSI$X25_USER is not defined on your system, your program must possess NETMBX privilege.

Note that to set up a virtual circuit requires certain system resources, which are deducted from the quota for your process. Refer to the *HP X.25 for OpenVMS—Programming Guide* for details.

**Format**

```
$QIO [efn],chan,func,[iosb],[astadr],[astprm], [p1], [p2],[p3],[p4],[p5],p6
```

**Arguments**

- **func**   IO$_ACCESS
- **p1**     Not used.
- **p2**     Starting virtual address of quadword descriptor of the NCB (see Appendix C).
- **p3**     Not used.
- **p4**     Not used.
- **p5**     Not used.
- **p6**     Unit number of the NV device, for X.29 programs. If the call is handled by the NW device (in X.25 programs), p6 must be zero. The default value for p6 is zero.

**NCB Contents**

Only mandatory, optional, and ignore items are listed in the following table. Other items will generate an error if you use them.
<table>
<thead>
<tr>
<th>PSISC_NCB Item Code</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mandatory items</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PVCNAM</td>
<td>PVC identifier</td>
<td>2,3</td>
</tr>
<tr>
<td><strong>Optional items</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALLED_EXTENSION</td>
<td>Called address extension</td>
<td>5</td>
</tr>
<tr>
<td>CALLING_EXTENSION</td>
<td>Calling address extension</td>
<td>5</td>
</tr>
<tr>
<td>CHARGING_INFO</td>
<td>Charging information request</td>
<td></td>
</tr>
<tr>
<td>CUG</td>
<td>(Bilateral) Closed User Group</td>
<td>1</td>
</tr>
<tr>
<td>CUM_TRST_DLY</td>
<td>Cumulative transit delay</td>
<td></td>
</tr>
<tr>
<td>DTECLASS</td>
<td>Name of the DTE Class from which a member DTE will be used to make the call</td>
<td>3,4</td>
</tr>
<tr>
<td>ETE_TRST_DLY</td>
<td>End–to–end transit delay</td>
<td></td>
</tr>
<tr>
<td>EXPEDITE</td>
<td>Negotiate use of interrupts</td>
<td></td>
</tr>
<tr>
<td>FSEL</td>
<td>Fast select (no restriction)</td>
<td></td>
</tr>
<tr>
<td>FSEL_RES</td>
<td>Fast select (restricted response)</td>
<td></td>
</tr>
<tr>
<td>LOCFAC</td>
<td>Local PSDN facilities</td>
<td></td>
</tr>
<tr>
<td>LOCSUBADR</td>
<td>Local subaddress ♦</td>
<td></td>
</tr>
<tr>
<td>MAX_TRST_DLY</td>
<td>Maximum acceptable transit delay</td>
<td></td>
</tr>
<tr>
<td>MIN_THRUCLS</td>
<td>Minimum throughput class (data rate)</td>
<td></td>
</tr>
<tr>
<td>NET_USER_ID</td>
<td>Network user identifier</td>
<td></td>
</tr>
<tr>
<td>PKTSIZE</td>
<td>Packet size</td>
<td></td>
</tr>
<tr>
<td>RCV_QUOTA</td>
<td>Maximum receive buffer bytes</td>
<td>3</td>
</tr>
<tr>
<td>REMDTE</td>
<td>Remote DTE address</td>
<td>1</td>
</tr>
<tr>
<td>REVCHG</td>
<td>Reverse charging request</td>
<td></td>
</tr>
<tr>
<td>RPOA</td>
<td>Remote Port of Access</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>Name of template created by network management containing specified parameters</td>
<td></td>
</tr>
<tr>
<td>THRUCLS</td>
<td>Throughput class (maximum data rate)</td>
<td></td>
</tr>
<tr>
<td>TRANSIT_DELAY</td>
<td>Maximum transit delay</td>
<td></td>
</tr>
<tr>
<td>userdata</td>
<td>User data field</td>
<td>6</td>
</tr>
<tr>
<td>WINSIZE</td>
<td>Window size</td>
<td></td>
</tr>
<tr>
<td><strong>Ignored items</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>Null item identifier</td>
<td></td>
</tr>
</tbody>
</table>
Notes

1. To specify the remote DTE for an SVC, you need to specify the DTE class using one of the following items:
   - DTECLASS
   - TEMPLATE (The template should contain a value for the DTE Class attribute)

In addition, you need to specify one of the following items:
   - REMDTE
   - CUG (if the CUG is not a BCUG, specify REMDTE also)
   - TEMPLATE (The template should contain a value for the Destination DTE Address attribute or the Selected Group attribute)

2. PVCNAM is used for X.25 programs only.

3. These are the only fields valid with a PVC.

4. The DTE Class could be a Remote DTE Class and may not have any member DTEs, for example, it could be used by an Access system to make calls through a Connector system.

5. The called address extension facility is encoded as follows:
   - Number of bytes in the facility (1 byte)
   - Number of semi–octets in the facility (1 byte)
   - The facility itself (up to 32 octets, with 2 digits per byte)

   Each of these bytes is encoded so that the low–order semi–octet is in bits 0 to 3, and the high–order semi–octet is in bits 4 to 7.

   When the matching is performed, a logical AND is performed on each byte of the facility with the corresponding byte of the mask and the result is compared with the corresponding byte of the value. The match succeeds if all the bytes compare. If the incoming call does not provide at least as many semi–octets as the extension value specifies, the match fails.

6. The user data field can be up to 16 bytes in length for normal calls and up to 128 bytes in length for fast select calls.
Examples

X.25 Code Example  In this example, the system service, IO$_ACCESS$, is called to set up a virtual circuit. The channel to the network device is PSI_CHAN. The I/O Status Block is declared as IO_STATUS, and the address of a descriptor of the NCB to be used is ACCESS_NCB.

; Declaring the data:
PSI_CHAN: .BLKW 1 ; Mailbox channel
IO_STATUS: .BLKW 4 ; I/O status block
ACCESS_NCB: .LONG ACCESS_NCB_LEN ; NCB descriptor
            .ADDRESS ACCESS_NCB_BLK

; Using the System Service:
$QIOW_S - ; Issue QIO and wait
        CHAN = PSI_CHAN,- ; to network device
        FUNC = #IO$_ACCESS,- ; Function is make call
        IOSB = IO_STATUS,- ; I/O status block
        P2 = #ACCESS_NCB ; Address of call NCB
                        ; descriptor
        P6 = 0 ; NV device unit number
                ; is zero for NW
**X.29 Code Example** In this example, the system service IO$_ACCESS$ is called to set up a virtual circuit for an X.29 call. The channel to the network device is NW_CHAN. The I/O Status Block is declared as IO_STATUS, and the address of a descriptor of the NCB to be used is ACCESS_NCB.

NW_CHAN:
  .BLKW 1 ; Channel to NW

NV_UNIT:
  .BLKL 1 ; NV Unit number

IO_STATUS:
  .BLKW 4 ; I/O Status block

;+ ; Network Connect Block:
;-
ACCESS_NCB: ; NCB Descriptor
  .LONG ACCESS_NCB_LEN
  .ADDRESS ACCESS_NCB_BLK

ACCESS_NCB_BLK: ; NCB to set up a call
  DTECLASS: ; DTE Class
    .WORD DTECLASS_LEN
    .WORD PSI$C_NCB_DTECLASS
    .ASCIC /ISO8208/
  DTECLASS_LEN = .-DTECLASS

REMOTE_DTE: ; DTE Address
  .WORD REMOTE_DTE_LEN
  .WORD PSI$C_NCB_REMDTE
  .ASCIC /23427341234522/
  REMOTE_DTE_LEN = .-REMOTE_DTE

; NO user data
; NO fast select
ACCESS_NCB_LEN = .-ACCESS_NCB_BLK

;+ ; Set up a virtual call
;-
$QIOW_S - ; Issue QIO and wait
  CHAN = NW_CHAN,- ; to network device
  FUNC = IO$_ACCESS,- ; function is make call
  IOSB = IO_STATUS,- ; I/O status block
  P2 = #ACCESS_NCB,- ; address of call NCB
  P6 = NV_UNIT ; NV unit number
  BS BW IO_ERROR ; Check for I/O error
Return Status

SS$_NORMAL  Service queued successfully (R0). Remote DTE has accepted the request to set up a virtual circuit (IOSB).

SS$_ACCVIO   Unable to read the NCB descriptor or the NCB.

SS$_CLEARED  The virtual circuit was cleared while this request was being processed. The remote DTE has rejected the request to set up a virtual circuit (IOSB).

SS$_DEVNOTMOUNT  X.25 for OpenVMS is not installed with X.29 support. NVDRIVER has not been installed by SYSGEN. X.29 calls cannot be made. This status is only returned for X.29 calls.

SS$_FILALRACC For SVCs: Invalid unit number for SVC already in use by another process.

For PVCs (X.25 only): The PVC is already being used by another process.

SS$_IVBUFLNEN The format of the NCB item–list is invalid. Check the secondary status in the third word of the IOSB. Details of the secondary status are given below.

SS$_IVDEVTMAM The format of the NCB is invalid, or an error was detected while processing the NCB. Check the secondary status in the IOSB. Details of the secondary status are given below.

SS$_IVLOGNAM  The logical name PSI$NETWORK has a length of 0 or has more than 63 characters.

SS$_NOLINKS  No internal logical channels available.

SS$_NOPRIV    The process does not have the privilege(s) required for this function.

SS$_NOSUCHNODE The PSDN specified in the NCB cannot be accessed from your system.

SS$_OPINCOMPL A previous call is still in progress on this channel.

SS$_RESULTOVF The translation of the logical name PSI$NETWORK has more than 16 characters.

Secondary Status Values

The secondary status values are found in the third word of the IOSB.

If the first word of the IOSB contains SS$_NORMAL, the third word can have one or more of the following flags set:

- PSI$M_STS_REMDTELNG  Remote DTE address too long — address truncated
- PSI$M_STS_PKBAD      Invalid packet size — nearest valid size chosen
- PSI$M_STS_RPOALNG    The length of the RPOA item is not a multiple of 4 digits, and has been truncated
- PSI$M_STS_THRBAD     Invalid throughput class — nearest valid class chosen
- PSI$M_STS_USERLNG    Too much user data supplied — data truncated
- PSI$M_STS_WINBAD     Invalid window size — nearest valid size chosen
- PSI$M_STS_WORDMBAD   The value of one of the transit delay items has been reduced to 65,535 ♦

If the first word of the IOSB contains SS$_ABORT or SS$_IVDEVTMAM, the third word can contain one of the status values shown in the following table.
<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSI$C_ERR_BADNAME</td>
<td>Bad counted string parameter. Correct and retry.</td>
</tr>
<tr>
<td>PSI$C_ERR_BADPARAM</td>
<td>Bad parameters specified. This may represent an internal error. If you are not able to find a parameter error, contact your local HP support representative for information about the variety of service options available to you and the procedures for submitting software problem reports.</td>
</tr>
<tr>
<td>PSI$C_ERR_CONFLICT</td>
<td>Conflicting items specified.</td>
</tr>
<tr>
<td>PSI$C_ERR_FACLNG</td>
<td>Facilities too long.</td>
</tr>
<tr>
<td>PSI$C_ERR_DTENOTAVAILABLE</td>
<td>The requested DTE is not available.</td>
</tr>
<tr>
<td>PSI$C_ERR_DTENOTINCLASS</td>
<td>The requested DTE is not a member of the specified DTE class.</td>
</tr>
<tr>
<td>PSI$C_ERR_DTENOTINGROUP</td>
<td>The requested DTE is not a member of the specified group.</td>
</tr>
<tr>
<td>PSI$C_ERR_INVEXP</td>
<td>Invalid use of expedited data negotiation.</td>
</tr>
<tr>
<td>PSI$C_ERR_INVITEM</td>
<td>Invalid item code.</td>
</tr>
<tr>
<td>PSI$C_ERR_INVNUM</td>
<td>Invalid ASCII number.</td>
</tr>
<tr>
<td>PSI$C_ERR_INVTRSTDLY</td>
<td>Invalid use of end–to–end transit delay facility; for example, MAX_TRST_DLY without ETE_TRST_DLY, or ETE_TRST_DLY without CUM_TRST_DLY.</td>
</tr>
<tr>
<td>PSI$C_ERR_L3ERR</td>
<td>Error returned from level 3.</td>
</tr>
<tr>
<td>PSI$C_ERR_MANYICI</td>
<td>More than one internal call identifier given.</td>
</tr>
<tr>
<td>PSI$C_ERR_NOACCESS</td>
<td>The X.25 Access module has been disabled or deleted.</td>
</tr>
<tr>
<td>PSI$C_ERR_NODTES</td>
<td>No DTE is available on which to make the call.</td>
</tr>
<tr>
<td>PSI$C_ERR_NOICI</td>
<td>No internal identifier specified.</td>
</tr>
<tr>
<td>PSI$C_ERR_NOL3</td>
<td>Internal error, contact your local HP support representative for information about the variety of service options available to you and the procedures for submitting software problem reports.</td>
</tr>
<tr>
<td>PSI$C_ERR_NOLOCAL</td>
<td>The ACP needs more logical workspace memory. Increase the nonpaged pool and retry.</td>
</tr>
<tr>
<td>PSI$C_ERR_NONONPAG</td>
<td>Not enough local workspace memory. Increase the nonpaged pool and retry.</td>
</tr>
<tr>
<td>PSI$C_ERR_NOSUCHDTE</td>
<td>The specified DTE is not known.</td>
</tr>
<tr>
<td>PSI$C_ERR_NOSUCHDTECLASS</td>
<td>The specified DTE class is not known.</td>
</tr>
<tr>
<td>PSI$C_ERR_NOSUCHGROUP</td>
<td>The specified group is not known.</td>
</tr>
<tr>
<td>PSI$C_ERR_NOSUCHPVC</td>
<td>The specified PVC is not known (X.25 only).</td>
</tr>
<tr>
<td>PSI$C_ERR_NOSUCHSECURITY</td>
<td>The security DTECLASS has not been found.</td>
</tr>
<tr>
<td>PSI$C_ERR_NOSUCHTEMPLATE</td>
<td>The specified template is not known.</td>
</tr>
<tr>
<td>PSI$C_ERR_NOTIMP</td>
<td>The requested feature is not yet implemented.</td>
</tr>
<tr>
<td>Code</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PSI$C_ERR_NOTRANS</td>
<td>No translation for this name (for example, unknown user group).</td>
</tr>
<tr>
<td>PSI$C_ERR_PVCALRACC</td>
<td>Internal error (X.25 only), contact your local HP support representative for information about the variety of service options available to you and the procedures for submitting software problem reports.</td>
</tr>
<tr>
<td>PSI$C_ERR_RECURLMT</td>
<td>Recursion limit reached, contact your local HP support representative for information about the variety of service options available to you and the procedures for submitting software problem reports.</td>
</tr>
<tr>
<td>PSI$C_ERR_UNKNOWN</td>
<td>Unspecified internal error, contact your local HP support representative for information about the variety of service options available to you and the procedures for submitting software problem reports.</td>
</tr>
</tbody>
</table>
2.7 $QIO(IO$_ACCESS!IO$M_ABORT) — Reject a Call

$QIO(IO$_ACCESS!IO$M_ABORT) rejects an incoming request to set up a virtual circuit.

If you subscribe to the fast select facility, you can use this call to send user data.

Note that you are advised to use the incoming call’s NCB as argument $p2$ to this QIO. Find the NCB in the mailbox associated with the channel that received the call. If you do not use the NCB as $p2$, the incoming call identifier must be copied from the incoming NCB.

If the rights identifier PSI$X25_USER is defined on your system, your program must possess either that rights identifier or BYPASS privilege.

If PSI$X25_USER is not defined on your system, your program must possess NETMBX privilege.

**Format**

$QIO [efn],chan,func,[iosb],[astadr],[astprm], [p1],[p2],[p3],[p4],[p5],[p6]

**Arguments**

- **func**  
  IO$_ACCESS!IO$M_ABORT
- **p1**  
  Not used.
- **p2**  
  Starting virtual address of quadword descriptor of the NCB (see Appendix C). See Note 1.
- **p3**  
  Not used.
- **p4**  
  Not used.
- **p5**  
  Not used.
- **p6**  
  Not used.

**NCB Contents:**

Only mandatory, optional, and ignore items are listed in the following table. Other items will generate an error if you use them.
<table>
<thead>
<tr>
<th>Item Code</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mandatory items</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICI</td>
<td>Incoming call identifier</td>
<td></td>
</tr>
<tr>
<td>Optional items</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALLED_EXTENSIONS</td>
<td>Called address extension</td>
<td></td>
</tr>
<tr>
<td>CAUSE</td>
<td>Code for PSDN clearing a call</td>
<td>2</td>
</tr>
<tr>
<td>DIAGCODE</td>
<td>Diagnostic code</td>
<td></td>
</tr>
<tr>
<td>LOCFACR</td>
<td>Local PSDN facilities</td>
<td></td>
</tr>
<tr>
<td>RESPDATA</td>
<td>Fast select response data</td>
<td>3</td>
</tr>
<tr>
<td>Ignored items</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDR_MOD_RSN</td>
<td>Reason for modifying line address</td>
<td></td>
</tr>
<tr>
<td>CALLING_EXTENSIONS</td>
<td>Calling address extension</td>
<td></td>
</tr>
<tr>
<td>CALL_REDIRECTED</td>
<td>Original DTE filter</td>
<td></td>
</tr>
<tr>
<td>CALL_REDIRECTED_REASON</td>
<td>Call redirection reason</td>
<td></td>
</tr>
<tr>
<td>CUG</td>
<td>(Bilateral) Closed User Group</td>
<td></td>
</tr>
<tr>
<td>CUM_TRST_DLY</td>
<td>Cumulative transit delay</td>
<td></td>
</tr>
<tr>
<td>DTECLASS</td>
<td>Name of the DTE Class from which a member DTE is used to make the call</td>
<td></td>
</tr>
<tr>
<td>ETE_TRST_DLY</td>
<td>End–to–end transit delay</td>
<td></td>
</tr>
<tr>
<td>EXPEDITE</td>
<td>Negotiate use of interrupts</td>
<td></td>
</tr>
<tr>
<td>FLT_PRI</td>
<td>Filter priority</td>
<td></td>
</tr>
<tr>
<td>FSEL</td>
<td>Fast select (no restriction)</td>
<td></td>
</tr>
<tr>
<td>FSEL_RES</td>
<td>Fast select (restricted response)</td>
<td></td>
</tr>
<tr>
<td>LOCFAC</td>
<td>Local PSDN facilities</td>
<td></td>
</tr>
<tr>
<td>LOCSUBADR</td>
<td>Local subaddress ♦</td>
<td></td>
</tr>
<tr>
<td>MAX_TRST_DLY</td>
<td>Maximum acceptable transit delay</td>
<td></td>
</tr>
<tr>
<td>MIN_THRUCLS</td>
<td>Minimum throughput class (data rate)</td>
<td></td>
</tr>
<tr>
<td>NET_USER_ID</td>
<td>Network user identifier</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>Null item identifier</td>
<td></td>
</tr>
<tr>
<td>PKTSIZE</td>
<td>Packet size</td>
<td></td>
</tr>
<tr>
<td>RCV QUOTA</td>
<td>Maximum number of receive buffers</td>
<td></td>
</tr>
<tr>
<td>REMDTE</td>
<td>Remote DTE address</td>
<td></td>
</tr>
<tr>
<td>REMSUBADR</td>
<td>Remote DTE subaddress</td>
<td></td>
</tr>
<tr>
<td>REVCHG</td>
<td>Reverse charging request</td>
<td></td>
</tr>
<tr>
<td>RPOA</td>
<td>Remote Port Of Access</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>Name of template created by network management containing specified parameters</td>
<td></td>
</tr>
<tr>
<td>Item Code</td>
<td>Meaning</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| THRUCLS      | Throughput class (maximum data rate) | ![Notes](1. You are advised to use the incoming call’s NCB as argument p2 to this QIO. Find the NCB in the mailbox associated with the channel that received the call. If you do not use the NCB as p2, the incoming call identifier must be copied from the incoming NCB.)
| TRANSIT_DELAY| Maximum transit delay            | ![Notes](2. This field is ignored unless X.25 for OpenVMS is operating as a DCE (Data Circuit–terminating Equipment) in order to connect to other DTEs outside the PSDN. X.25 for OpenVMS can operate as a DCE to connect back–to–back with another DTE. DTEs cannot use nonzero codes, but DCEs can.)
| USERDATA     | User data field                  | ![Notes](3. Use this field only for fast select calls.)               |
| WINSIZE      | Window size                      |                                                                       |
Example
In the following example, IO$_ACCESS!IO$M_ABORT rejects the request. The channel to the network device is PSI_CHAN and the NCB descriptor begins at ABORT_NCB.

; Declaring the data:
PSI_CHAN: .BLKW 1 ; Mailbox channel
ABORT_NCB: .BLKQ 1 ; NCB descriptor

; Using the System Service:
$QIOW_S - ; Issue QIO and wait
  CHAN = PSI_CHAN,- ; to the network device
  FUNC = #IO$_ACCESS!IO$M_ABORT,- ; Function is reject
    ; request
  P2 = #ABORT_NCB ; NCB descriptor address

Return Status
SS$_NORMAL Service successfully queued (R0). Request successfully rejected (IOSB).
SS$_ACCVIO Unable to read argument p2.
SS$_CLEARED The virtual circuit was cleared while this request was being processed, or the circuit was in the process of being cleared when you issued the request.
SS$_FILALRACC Invalid unit number for SVC.
SS$_IVBUFLEN The format of the NCB item–list is invalid. See the secondary status in the IOSB.
SS$_IVDEVNAM The format of the NCB is invalid, or error detected while processing the NCB. See the secondary status in the IOSB.
SS$_OPINCOMPL A previous call is still in progress on this channel.
2.8 $QIO(IO$_ACCESS!!O$M_ACCEPT) — Accept a Call

$QIO(IO$_ACCESS!!O$M_ACCEPT) accepts an incoming request from a remote DTE (X.25) or a remote PAD (X.29) to set up a virtual circuit.

The parameters requested in the incoming call can be negotiated using this QIO function. Parameter negotiation can be achieved either by specifying individual items to be negotiated or by specifying the items in a template (see NCB TEMPLATE item code). Note that if a template is not defined, the Default template is used.

A template can also be used to supply parameters that are not defined in the NCB used to accept the call.

If you subscribe to the fast select acceptance facility, you can use this call to send user data.

Note that you are advised to use the NCB received as argument p2. Otherwise, you should copy the incoming call identifier (PSI$C_NCB_ICI) from the received NCB. Read the NCB in the mailbox associated with the channel that received the call.

If the rights identifier PSI$X25_USER is defined on your system, your program must possess either that rights identifier or BYPASS privilege.

If PSI$X25_USER is not defined on your system, your program must possess NETMBX privilege.

Note that to accept a request to set up a virtual circuit requires certain system resources which are deducted from the quota for your process. See the HP X.25 for OpenVMS—Programming Guide for details.

**Format**

$QIO [efn],chan,func,[iosb],[astadr],[astprm], [p1], p2, [p3],[p4],[p5],[p6]

**Arguments**

- **func**  
  IO$_ACCESS!!O$M_ACCEPT.
- **p1**  
  Not used.
- **p2**  
  Starting virtual address of quadword descriptor of the NCB (see Appendix C).

  You are advised to use the incoming call’s NCB as argument p2 to this QIO. Find the NCB in the mailbox associated with the channel that received the call. If you do not use the incoming NCB, you must copy the incoming call destination from that NCB.

- **p3**  
  Not used.
- **p4**  
  Not used.
- **p5**  
  Not used.
- **p6**  
  Unit number of the NV device (zero for X.25 programs).

**NCB Contents:**

Only mandatory, optional, and ignored items are listed in the following table. Other items will generate an error if you use them.
<table>
<thead>
<tr>
<th>Item Code</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICI</td>
<td>Incoming call identifier</td>
<td></td>
</tr>
<tr>
<td><strong>Mandatory items</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALLED_EXTENSION</td>
<td>Called address extension</td>
<td>1</td>
</tr>
<tr>
<td>CUM_TRST_DLY_R</td>
<td>Cumulative transit delay</td>
<td></td>
</tr>
<tr>
<td>EXPEDITE</td>
<td>Negotiate use of interrupts</td>
<td></td>
</tr>
<tr>
<td>LOCFACR</td>
<td>Local PSDN facilities</td>
<td></td>
</tr>
<tr>
<td>NET_USER_ID</td>
<td>Network user identifier</td>
<td></td>
</tr>
<tr>
<td>RESPDATA</td>
<td>Fast select response data</td>
<td>2</td>
</tr>
<tr>
<td>PKTSIZE</td>
<td>Packet size</td>
<td></td>
</tr>
<tr>
<td>RCV_QUOTA</td>
<td>Maximum number of receive buffers</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>Name of template created by network management</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>containing specified parameters</td>
<td></td>
</tr>
<tr>
<td>THRUCLS</td>
<td>Throughput class (maximum data rate)</td>
<td></td>
</tr>
<tr>
<td>WINSIZE</td>
<td>Window size</td>
<td></td>
</tr>
<tr>
<td><strong>Optional items</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDR_MOD_RSN</td>
<td>Reason for modifying line address</td>
<td></td>
</tr>
<tr>
<td>CALLING_EXTENSION</td>
<td>Calling address extension</td>
<td></td>
</tr>
<tr>
<td>CALL_RedIR_ORIG</td>
<td>Original DTE filter</td>
<td></td>
</tr>
<tr>
<td>CALL_RedIR_RSN</td>
<td>Call redirection reason</td>
<td></td>
</tr>
<tr>
<td>CUG</td>
<td>(Bilateral) Closed User Group</td>
<td></td>
</tr>
<tr>
<td>CUM_TRST_DLY</td>
<td>Cumulative transit delay</td>
<td></td>
</tr>
<tr>
<td>ETE_TRST_DLY</td>
<td>End–to-end transit delay</td>
<td></td>
</tr>
<tr>
<td>FSEL</td>
<td>Fast select (no restriction)</td>
<td></td>
</tr>
<tr>
<td>FSEL_RES</td>
<td>Fast select (restricted response)</td>
<td></td>
</tr>
<tr>
<td>LOCFAC</td>
<td>Local PSDN facilities</td>
<td></td>
</tr>
<tr>
<td>LOCSUBADR</td>
<td>Local subaddress</td>
<td>♦</td>
</tr>
<tr>
<td>MAX_TRST_DLY</td>
<td>Maximum acceptable transit delay</td>
<td></td>
</tr>
<tr>
<td>MIN_THRUCLS</td>
<td>Minimum throughput class (data rate)</td>
<td></td>
</tr>
<tr>
<td>DTECLASS</td>
<td>Name of a DTE Class from which a member DTE is used</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to make the call</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>Null item identifier</td>
<td></td>
</tr>
<tr>
<td>FLT_PRI</td>
<td>Filter priority</td>
<td></td>
</tr>
<tr>
<td>Item Code</td>
<td>Meaning</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>REMDTE</td>
<td>Remote DTE address</td>
<td></td>
</tr>
<tr>
<td>REMSUBADR</td>
<td>Remote DTE subaddress</td>
<td></td>
</tr>
<tr>
<td>REVCHG</td>
<td>Reverse charging request</td>
<td></td>
</tr>
<tr>
<td>RPOA</td>
<td>Remote Port Of Access</td>
<td></td>
</tr>
<tr>
<td>TRANSIT_DELAY</td>
<td>Maximum transit delay</td>
<td></td>
</tr>
<tr>
<td>USERDATA</td>
<td>User data field</td>
<td></td>
</tr>
</tbody>
</table>

**Notes**

1. The called address extension facility is encoded as follows:
   - Number of bytes in the facility (1 byte)
   - Number of semi-octets in the facility (1 byte)
   - The facility itself (up to 32 octets, with 2 digits per byte)
     Each of these bytes is encoded so that the low-order semi-octet is in bits 0 to 3, and the high-order semi-octet is in bits 4 to 7.

   When the matching is performed, a logical AND is performed on each byte of the facility with the corresponding byte of the mask and the result is compared with the corresponding byte of the value. The match succeeds if all the bytes compare. If the incoming call does not provide at least as many semi-octets as the extension value specifies, the match fails.

2. Use this field for fast select calls only.

3. Include all parameters in the template that are to be negotiated or that are to be supplied if not present in the NCB used to accept the call. If the TEMPLATE item code is not defined, the Default template is used.
Examples

In the following example, the request to set up a virtual circuit is accepted with the IO$ACCESS!IO$M_ACCEPT call. The channel to the network device is PSI_CHAN. An I/O status block, IO_STATUS, is to receive the completion status, and the starting address of the descriptor of the NCB is ACCEPT_NCB.

; Declaring the data:
PSI_CHAN: .BLKW 1 ; Mailbox channel
IO_STATUS: .BLKW 4 ; I/O status block
ACCEPT_NCB: .BLKQ 1 ; NCB descriptor

; Using the System Service:
$QIOW_S - ; Issue QIO and wait
CHAN = PSI_CHAN,- ; to the network device
FUNC = #IO$ACCESS!IO$M_ACCEPT,- ; Function is accept call
IOSB = IO_STATUS,- ; I/O status block
P2 = ACCEPT_NCB ; NCB descriptor address

In the following example, the request to set up a virtual circuit is accepted with the IO$ACCESS!IO$M_ACCEPT call. The channel to the network device is NW_CHAN. NV_UNIT supplies the NV device number. An I/O status block, IO_STATUS, is to receive the completion status, and the starting address of the descriptor of the NCB is ACCEPT_NCB.

NW_CHAN: .BLKW 1 ; Channel to NW
NV_UNIT: .BLKL 1 ; NV Unit number
IO_STATUS: .BLKW 4 ; IO Status block

; Network Connect Block:
ACCEP_NCB: ; NCB Descriptor
.BLKL 2

; Accept a virtual call
$QIOW_S - ; Issue QIO and wait
CHAN = NW_CHAN,- ; to network device
FUNC = #IO$ACCESS!IO$M_ACCEPT,- ; function is accept call
IOSB = IO_STATUS,- ; I/O status block
P2 = #ACCESS_NCB,- ; address of call NCB
P6 = NV_UNIT ; NV unit number
BSBW IO_ERROR ; Check for I/O error
### Return Status

<table>
<thead>
<tr>
<th>Return Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS$ _NORMAL</td>
<td>Service queued successfully (R0). The virtual circuit has been accepted (IOSB).</td>
</tr>
<tr>
<td>SS$ _ACCVIO</td>
<td>Unable to read argument p2.</td>
</tr>
<tr>
<td>SS$ _CLEARED</td>
<td>The virtual circuit was cleared while this request was being processed, or the circuit was in the process of being cleared when you issued the request.</td>
</tr>
<tr>
<td>SS$ _DEVOFFLINE</td>
<td>Local DTE or X.25 Connector node is being shut down.</td>
</tr>
<tr>
<td>SS$ _FILALRACC</td>
<td>Invalid unit number for SVC already in use by another process.</td>
</tr>
<tr>
<td>SS$ _FILNOTACC</td>
<td>No SVC on the specified channel to NV (X.29 only).</td>
</tr>
<tr>
<td>SS$ _IVDEVNAM</td>
<td>The format of the NCB is invalid, or an error was detected while processing the NCB. Check the secondary status in the IOSB.</td>
</tr>
<tr>
<td>SS$ _NOSUCHNODE</td>
<td>The incoming call you are trying to accept no longer exists; or, the incoming call identifier (PSICC_NCB_ICI) was incorrect; or, the call is using the facility for fast select (restricted response).</td>
</tr>
<tr>
<td>SS$ _OPINCOMPL</td>
<td>A previous call is still in progress on this channel.</td>
</tr>
</tbody>
</table>
2.9 $QIO(IO$_ACCESS!IO$M_REDIRECT) — Redirect a Call

$QIO(IO$_ACCESS!IO$M_REDIRECT) redirects an incoming call request to another process before the call request is accepted or rejected.

The call uses the subaddress and other addressing information specified in the NCB in the normal way, to associate the new process with the call.

The system service completes when X.25 for OpenVMS redirects the incoming call request.

Note that you are advised to use the NCB received as argument $p2$. Otherwise, you should copy the incoming call identifier (PSI$C_NCB_ICI) from the received NCB. Read the NCB in the mailbox associated with the channel that received the call.

To ensure that your process is not searched again, move the priority value from PSI$C_NCB_FLT_PRI to PSI$C_NCB_FLT_REDPRI.

If the rights identifier PSI$X25_USER is defined on your system, your program must possess either that rights identifier or BYPASS privilege.

If PSI$X25_USER is not defined on your system, your program must possess NETMBX privilege.

**Format**

$\textit{QIO}\{efn,\textit{chan,func,iosb,astadr,astprm, p1,p2,p3,p4,p5,p6}\}$

**Arguments**

- **func** \(\text{IO$_ACCESS!IO$M_REDIRECT}\).
- **p1** Not used.
- **p2** Starting virtual address of quadword descriptor of the NCB.
  
  You are advised to use the incoming call's NCB as argument $p2$ to this QIO. Find the NCB in the mailbox associated with the channel that received the call. If you do not use the NCB as $p2$, the incoming call identifier must be copied from the incoming NCB (see Appendix C for further details).
- **p3** Not used.
- **p4** Not used.
- **p5** Not used.
- **p6** Not used.

**NCB Contents:**

Only mandatory, optional, and ignored items are listed in the following table. Other items will generate an error if you use them.
<table>
<thead>
<tr>
<th>Item Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mandatory items</strong></td>
<td></td>
</tr>
<tr>
<td>ICI</td>
<td>Incoming call identifier.</td>
</tr>
<tr>
<td>FILTER</td>
<td>Name of filter to which to redirect the call. Note that the filter</td>
</tr>
<tr>
<td></td>
<td>to which you redirect the call is not used to rematch the call</td>
</tr>
<tr>
<td>FLT_REDPRI</td>
<td>Redirection priority.</td>
</tr>
<tr>
<td><strong>Optional items</strong></td>
<td></td>
</tr>
<tr>
<td>CALLED_EXTENSION</td>
<td>Called address extension</td>
</tr>
<tr>
<td>CALL_REDIR_ORIG</td>
<td>Original DTE filter</td>
</tr>
<tr>
<td>CALL_REDIR_RSN</td>
<td>Call redirection reason</td>
</tr>
<tr>
<td>LOCSUBADR</td>
<td>Local subaddress ♦</td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>Name of template created by network management with specified parameters</td>
</tr>
<tr>
<td>USERDATA</td>
<td>User data field</td>
</tr>
<tr>
<td><strong>Ignored items</strong></td>
<td></td>
</tr>
<tr>
<td>ADDR_MOD_RSN</td>
<td>Reason for modifying line address</td>
</tr>
<tr>
<td>CALLING_EXTENSION</td>
<td>Calling address extension</td>
</tr>
<tr>
<td>CUG</td>
<td>(Bilateral) Closed User Group</td>
</tr>
<tr>
<td>CUM_TRST_DLY</td>
<td>Cumulative transit delay</td>
</tr>
<tr>
<td>ETE_TRST_DLY</td>
<td>End–to–end transit delay</td>
</tr>
<tr>
<td>EXPEDITE</td>
<td>Negotiate use of interrupts</td>
</tr>
<tr>
<td>FSEL</td>
<td>Fast select (no restriction)</td>
</tr>
<tr>
<td>FSEL_RES</td>
<td>Fast select (restricted response)</td>
</tr>
<tr>
<td>LOCFAC</td>
<td>Local PSDN facilities</td>
</tr>
<tr>
<td>MAX_TRST_DLY</td>
<td>Maximum acceptable transit delay</td>
</tr>
<tr>
<td>MIN_THRUCLS</td>
<td>Minimum throughput class (data rate)</td>
</tr>
<tr>
<td>DTECLASS</td>
<td>Name of a DTE Class from which a member DTE is used to make the call</td>
</tr>
<tr>
<td>NULL</td>
<td>Null item identifier</td>
</tr>
<tr>
<td>PKTSIZE</td>
<td>Packet size</td>
</tr>
<tr>
<td>FLT_PRI</td>
<td>Filter priority</td>
</tr>
<tr>
<td>RCV_QUOTA</td>
<td>Maximum number of receive buffers</td>
</tr>
<tr>
<td>REMDTE</td>
<td>Remote DTE address</td>
</tr>
<tr>
<td>REMSUBADR</td>
<td>Remote DTE subaddress</td>
</tr>
<tr>
<td>REVCHG</td>
<td>Reverse charging request</td>
</tr>
<tr>
<td>RPOA</td>
<td>Remote Port Of Access</td>
</tr>
<tr>
<td>PSISC_NCB Item Code</td>
<td>Meaning</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Ignored items</td>
<td></td>
</tr>
<tr>
<td>THRUCLS</td>
<td>Throughput class (maximum data rate)</td>
</tr>
<tr>
<td>TRANSIT_DELAY</td>
<td>Maximum transit delay</td>
</tr>
<tr>
<td>WINSIZE</td>
<td>Window size</td>
</tr>
</tbody>
</table>

**Example**

In this example the QIO is used with function code IO$ACCESS!IO$M_REDIRECT, to redirect the call according to the parameters in the NCB REDIRECT_NCB.

```plaintext
; Declaring the data:
PSI_CHAN:
    .BLKW 1 ; Mailbox channel
IO_STATUS:
    .BLKW 1 ; I/O status block
REDIRECT_NCB:
    .BLKQ 1 ; NCB descriptor

; Using the System Service:
$QION_S - ; Issue QIO and wait
    CHAN = PSI_CHAN,- ; to the network device
    FUNC = #IO$_ACCESS!IO$M_REDIRECT,- ; Function is
    IOSB = IO_STATUS,- ; I/O status block
    P2 = REDIRECT_NCB ; NCB descriptor address
```

**Return Status**

- **SS$_NORMAL** Service queued successfully (R0). Request redirected successfully (IOSB).
- **SS$_ACCVIO** Unable to read argument p2.
- **SS$_IVBUFLEN** The format of the NCB item–list is invalid. See the secondary status in the IOSB.
- **SS$_IVDEVNAM** Either the format of the NCB is invalid, or an error was detected while processing the NCB. Check the secondary status in the IOSB.
- **SS$_NOSUCHNODE** Either the incoming call you are trying to redirect no longer exists, or the incoming call identifier (PSISC_NCB_ICI) is incorrect.
- **SS$_OPINCOMPL** A previous call is still in progress on this channel.
- **SS$_TOOMANYREDS** This call has been redirected more than eight times, and cannot be redirected again. ♦
2.10 \$QIO(IO$_ACPCONTROL) — Declare a Network Process

\$QIO(IO$_ACPCONTROL) allows the current process to select which incoming calls it handles.

The network process declaration block specifies the matching parameters for the filter. Incoming calls are placed in the mailbox associated with the channel over which this \$QIO(IO$_ACPCONTROL) was issued. The calls are selected according to the following criteria:

- The filter parameters match those of the call.
- Either the filter has the highest priority of those matching the call, or the call has been redirected from a filter with a priority greater than or equal to that of the current filter.

The request can specify a number of parameters to identify the incoming calls it will handle, and these parameters form an X25 ACCESS FILTER entity for your process. These parameters are shown in Table 2–2.

The filter specified by \$QIO(IO$_ACPCONTROL) can be one of two types:

- Static
  This type of filter is one that is created using management commands. It is available until either disabled or deleted.

- Dynamic
  This type of filter is created dynamically by defining its characteristics in the \$QIO(IO$_ACPCONTROL) call. A filter created in this way ceases to exist when the specified channel is deassigned.

The action taken when the \$QIO(IO$_ACPCONTROL) call is processed depends on the filter name specified in the call:

- If the filter name matches a static filter, that filter is used to listen for incoming calls.

- If the filter name does not match a static filter, a dynamic filter having the filter characteristics specified in the \$QIO(IO$_ACPCONTROL) call is created and used to listen for incoming calls.

If a filter name is not specified in the \$QIO(IO$_ACPCONTROL) call, one of the following default filters is created:

- If the system service call is an X.25 call (the ACCLVL item code contains the string “X25L3”), a filter having a name of the following format is created:
  
  \textit{X25L3-pid-xxxx}

  where:

  \begin{align*}
  \text{\textit{pid}} & \quad \text{is the process identifier (in hex)} \\
  \text{\textit{xxxx}} & \quad \text{is a unique number (in hex)}
  \end{align*}
• If the system service call is an X.29 call (the ACCLVL item code contains the string “X29”), a filter having a name of the following format is created:

\[
\text{X29-} \text{pid-} \text{xxxx}
\]

where:

\[
\begin{align*}
\text{pid} & \quad \text{is the process identifier (in hex)} \\
\text{xxxx} & \quad \text{is a unique number (in hex)}
\end{align*}
\]

Further filters can be added by issuing more than one QIO with a function code of IO$_\text{ACPCONTROL}$, but you cannot change the information associated with an existing filter.

Filters can be deleted without stopping your process, by deassigning the channel over which $QIO(IO$_\text{ACPCONTROL})$ was issued.

For information about displaying the filters in the X25 Access module, see HP X.25 for OpenVMS—Management Guide.

All incoming calls whose parameters match those specified in this QIO are placed in the mailbox associated with the channel over which this QIO request was issued.

Details of the network process declaration block contents are given below.

**Format**

\[
\text{$QIO \{efn,chan,func, [iosb],[astadr],[astprm], p1,p2,[p3],[p4], [p5],[p6]\}}
\]

**Arguments**

\[
\begin{align*}
\text{func} & \quad \text{IO$_\text{ACPCONTROL}$} \\
\text{p1} & \quad \text{Address of the quadword descriptor of a 5–byte block. The format of the 5–byte block is:} \\
& \text{\quad .BYTE NFB$_\text{C_DECLNAME}$} \\
& \text{\quad .LONG 0} \\
& \text{\quad The contents of the 5–byte block are:} \\
& \text{\quad A function type code (one byte)} \\
& \text{\quad A longword parameter value} \\
& \text{\quad The function type is a symbol defined by the $NFBDEF$ macro in the library LIB.MLB.} \\
\text{p2} & \quad \text{Starting virtual address of the quadword descriptor of a network process declaration block (see below).} \\
\text{p3} & \quad \text{Not used.} \\
\text{p4} & \quad \text{Not used.} \\
\text{p5} & \quad \text{Not used.} \\
\text{p6} & \quad \text{Not used.}
\end{align*}
\]

**Network Process Declaration Block**

The network process declaration block consists of items of information. The items are of variable length, each item containing the following fields:

- **Word 1:** Length of data in this item, including the length and type fields.
- **Word 2:** Type code, of the form PSI$_\text{C_NTD_code}$.
- **Subsequent words:** Data (variable length).
Each type of data has a specific format: single-byte value, counted string, or longword value. Table 2–2 summarizes the permitted type codes.

Table 2–2 Item Codes for a Network Process Declaration Block

<table>
<thead>
<tr>
<th>PSISC_NTDL Item Code</th>
<th>Meaning</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mandatory item codes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCLVL</td>
<td>Access level</td>
<td>Counted ASCII string, which declares the filter of either:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• An X25 listener (ACCLVL contains the string “X25L3”)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• An X29 listener (ACCLVL contains the string “X29”)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See Note 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Optional item codes</strong></td>
</tr>
<tr>
<td>CALLED_DTE</td>
<td>Called DTE address</td>
<td>Counted string, containing the address of the DTE originally called. This item is used after a call has been redirected.</td>
</tr>
<tr>
<td>DATMSK</td>
<td>Data mask</td>
<td>Counted string of up to 16 bytes. X.25 for OpenVMS performs a logical AND operation between the mask and the user data of the incoming call; the result of this operation is compared with the data specified in PSISC_NTDL_USRDATA. If PSI$C_NTDL_DATMSK is not specified, no comparison is made.</td>
</tr>
<tr>
<td>EXTMSK</td>
<td>Called extension</td>
<td>Counted string containing a byte that specifies the number of semi–octets in the mask, followed by the mask value of up to 40 semi–octets. X.25 for OpenVMS performs a logical AND operation between the mask and the PSI$C_NCB_CALLED_EXTENSION field of the incoming call; the result of this operation is compared with the data specified in PSI$C_NTDL_EXTVAL. If PSI$C_NTDL_EXTMSK is not specified, no comparison is made.</td>
</tr>
<tr>
<td>EXTVAL</td>
<td>Called extension</td>
<td>Counted string containing a byte that specifies the number of semi–octets in the value, followed by the value of up to 40 semi–octets. X.25 for OpenVMS compares the data with the incoming data masked by PSI$C_NTDL_EXTMSK. If PSI$C_NTDL_EXTVAL is not specified, no comparison is made.</td>
</tr>
<tr>
<td>INCIDENT</td>
<td>Incoming DTE</td>
<td>Counted ASCII string, containing the Called DTE field from an incoming call packet.</td>
</tr>
</tbody>
</table>

(continued on next page)
Table 2–2 (Cont.) Item Codes for a Network Process Declaration Block

<table>
<thead>
<tr>
<th>Item Code</th>
<th>Meaning</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSI$C_NTD</td>
<td>Optional item codes</td>
<td>Counted ASCII string, specifying the name of:</td>
</tr>
</tbody>
</table>
| FILTER    | Name of the filter in the X25 Access module | • A static filter created by network management.  
• A dynamic filter created by $QIO(IO$_ACPCONTROL) with the specified filter name. If FILTER is not specified, then the filter name is derived from the ACCLVL item code (see the call description). |
| DTECLASS  | DTE Class | Counted ASCII string. |
| FLT_PRI   | Priority | Single–word value, in the range 0 (low) to 65,535 (high). The default priority is 3000. |
| RCVDTE    | Receiving DTE | Counted ASCII string, containing the address of the local DTE that received the call. |
| REDRSN    | Redirect reason | 32–bit integer, containing one of the following symbolic values: |
|           |          | • PSI$C_REDRSN_BUSY  
• PSI$C_REDRSN_OUT_OF_ORDER  
• PSI$C_REDRSN_SYSTEMATIC |
| REMDTE    | Remote DTE | Counted ASCII string. If not specified, all remote DTE addresses are handled. |
| SAHI      | Subaddress high | 16–bit integer, containing the highest value of the range of subaddresses to be handled. If you do not specify a range, all subaddresses are handled. ♦ |
| SALO      | Subaddress low | 16–bit integer, containing the lowest value of the range of subaddresses to be handled. If you do not specify a range, all subaddresses are handled. See Note 5. ♦ |
| USRDATA   | User data | Counted ASCII string of up to 16 bytes. X.25 for OpenVMS compares the user data with the incoming data masked by PSI$C_NTD_DATMSK. If USRDATA is not specified, no comparison is made. |
| USRGRP    | Closed user group | Counted ASCII string. If not specified, all closed user groups are handled. |
Notes

1. If ACCLVL = X25L3, you must accept, reject or redirect the call whose details are in the NCB.
   If ACCLVL = X29, the call has been accepted by X.25 for OpenVMS, and you must assign a channel to the NV device in the mailbox message.

2. When you specify user data and a data mask, the network process performs a logical AND operation between the mask and the user data field of the incoming call. The network process then compares the result of this operation with the user data value contained in the network process declaration block, and accepts the call only if the values match.

3. The called address extension facility is encoded as follows:
   - Number of bytes in the facility (1 byte)
   - Number of semi-octets in the facility (1 byte)
   - The facility itself (up to 32 octets, with 2 digits per byte)
     Each of these bytes is encoded so that the low-order semi-octet is in bits 0 to 3, and the high-order semi-octet is in bits 4 to 7.
     When the matching is performed, a logical AND operation is performed between each byte of the facility and the corresponding byte of the mask and the result is compared with the corresponding byte of the value. The match succeeds if all the bytes compare. If the incoming call does not provide at least as many semi-octets as the extension value specifies, the match fails.

4. The values for the redirect reason code are as follows:

<table>
<thead>
<tr>
<th>Symbolic Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSI$C_REDRSN_BUSY</td>
<td>Busy</td>
</tr>
<tr>
<td>PSI$C_REDRSN_OUT_OF_ORDER</td>
<td>Out of order</td>
</tr>
<tr>
<td>PSI$C_REDRSN_SYSTEMATIC</td>
<td>Automatic redirection</td>
</tr>
</tbody>
</table>

5. If you want a single subaddress or a range of subaddresses, you must specify both the lowest and highest values. For a single subaddress, specify the same value for the SAHI and SALO fields. If you specify a subaddress range, only calls specifying a subaddress are handled (even if you specify the full range of 0 to 99). If you do not specify a range, all subaddresses are handled.

   To filter for specific incoming calls, use the Incoming DTE Address attribute of the X25 ACCESS FILTER entity, or use the optional PSI$C_NTD_INCDTE item in the Network Process Declaration block.

6. If the rights identifier PSI$DECLNAME is defined on your system, your process must possess either that rights identifier or BYPASS privilege.
   If PSI$DECLNAME is not defined on your system, your process must possess NETMBX privilege.
Examples

X.25 Code Example  In the following example, the 5–byte function control block is at address FUNC_BUF. The descriptor of the network process declaration block is at address DEC_BUF. The channel to the network device is PSI_CHAN.

; Declaring the data:
PSI_CHAN:
   .BLKW 1 ; Mailbox channel
IO_STATUS:
   .BLKQ 1 ; I/O status block
FUNC_DESC:
   .LONG 5 ; Function block descriptor
   ADDRESS FUNC_BUF ; Address
FUNC_BUF:
   .BYTE NFB$C_DECLNAME
   .LONG 0
DEC_DESC:
   .LONG DEC_BUF_LEN ; Length
   ADDRESS DEC_BUF ; Address
DEC_BUF:
   ; Declaration block
ACCLVL:
   .WORD ACCLVL_LEN ; Length field
   .WORD PSI$C_NTD_ACCLVL ; Type field
   .ASCIC 'X25L3' ; String
ACCLVL_LEN = .-ACCLVL
; Item list to match called address extension of A15
EXTMSK:
   .WORD EXTMSK_LEN ; Item length
   .WORD PSI$C_NTD_EXTMSK ; Item type
   .BYTE EXTMSK_COUNT ; Byte count for "counted string"
EXTMSK_STR:
   .BYTE 3 ; Semi-octet count
   .BYTE ^FF ; First two semi-octets
   .BYTE ^XF0 ; Third semi-octet
EXTMSK_COUNT = .-EXTMSK_STR
EXTMSK_LEN = .-EXTMSK
EXTVAL:
   .WORD EXTVAL_LEN ; Item length
   .WORD PSI$C_NTD_EXTVAL ; Item type
   .BYTE EXTVAL_COUNT ; Byte count for "counted string"
EXTVAL_STR:
   .BYTE 3 ; Count in semi-octets
   .BYTE ^XA1 ; First two semi-octets
   .BYTE ^X50 ; Third semi-octet
EXTVAL_COUNT = .-EXTVAL_STR
EXTVAL_LEN = .-EXTVAL
DEC_BUF_LEN = .-DEC_BUF
; Using the System Service:
$QIOW_S - ; Issue QIO and wait
CHAN = PSI_CHAN,- ; to the network device
FUNC = #IO$_ACP$CONTROL- ; Function is ACP control
IOSB = IO_STATUS,- ; I/O status block
P1 = FUNC_DESC,- ; Function descriptor address
P2 = #DEC_DESC ; Process declaration address
**X.29 Code Example**  In this example, the QIO with function code IO$_\_ACPCONTROL is used to tell X.25 for OpenVMS to accept all X.29 calls. The 5-byte function control block is at address DECL_BUF. The descriptor of the network process declaration block is at address NPDB_DESC. The channel to the network device is NW_CHAN.

```
NW_CHAN:
  .BLKW 1 ; Channel to NW
DECL_DESC:
  .LONG 5 ; Five byte buffer
  .ADDRESS DECL_BUF ; Address of buffer
DECL_BUF:
  .BYTE NFB$C_DECLNAME ; function type
  .LONG 0 ; parameter
NPDB_DESC:
  ; Network Process Declaration
  ; Buffer descriptor
  .LONG NPDB_BUF_LEN ; Size of buffer
  .ADDRESS NPDB_BUF ; Address of buffer
NPDB_BUF:
NPD_ITEM1:
  .WORD NPD_ITEM1_LEN ; Length of first item
  .WORD PSI$C_NTD_ACCLVL ; Access Level
  .ASCII /X29/ ; ASCII counted string
NPD_ITEM1_LEN = . - NPD_ITEM1 ; Calculate the size in words
NPD_BUF_LEN = . - NPDB_BUF

; Tell X.25 that we want to accept all X29 calls
;
$QIOW_S - ; Send a QIO and wait
  IOSB = IO_STATUS,- ; I/O status block
  CHAN = NW_CHAN,- ; NW channel
  FUNC = #IO$_ACPCONTROL,- ; function is write
  P1 = DECL_DESC,- ; descriptor of declname block
  P2 = #NPDB_DESC ; descriptor of declaration block
  BSBW IO_ERROR ; Check system service and IOSB
```
Return Status

SS$_NORMAL  Service completed successfully. The filter has been added to the X25 Access module.

SS$_ACCVIO  The program cannot read the descriptor for argument p1 or p2; or the item–list in the Network Process Declaration Block; or the 5-byte block.

SS$_BADPARAM  A logically incorrect item has been specified in the Network Process Declaration Block. For example, you have attempted to put the item code PSI$C_NTDFLT_PRI in the Network Process Declaration Block when the filter name (specified using the PSI$C_NTD_FILTER item code) is associated with a static filter.

SS$_DIRFULL  No more room for declared name.

SS$_ILLCNTRFUNC  The first byte of the 5-byte block must be NFB$C_DECLNAME, and the rest of the block must be zeroed.

SS$_IVDEVNAM  The format of the NCB is invalid, or an error was detected while processing the NCB. See the secondary status in the IOSB.

SS$_NOMBX  There is no mailbox associated with the channel on which $QIO(IO$_ACPCONTROL) was issued.

Secondary Status Values

PSI$C_ERR_FILTERALREADYSET  The named filter has already been claimed.

PSI$C_ERR_NOACCESS  The X.25 Access module has been disabled or deleted.

PSI$C_ERR_NOL3  Internal error. Contact your local HP support representative for information about the variety of service options available to you and the procedures for submitting software problem reports.
2.11 $QIO(IO$_DEACCESS) — Clear a Virtual Circuit

The QIO system service with a function code of IO$_DEACCESS clears a
switched virtual circuit or confirms receipt of a call cleared message. For PVCs,
this call closes the virtual circuit so that the PVC can be reassigned to another
process.

Note that clearing a virtual circuit can result in loss of data in either direction
(see the HP X.25 for OpenVMS—Programming Guide for details).

**Format**

$QIO [efn],chan,func,[iosb],[astadr],[astprm], [p1],[p2],[p3],[p4],[p5],p6

**Arguments**

- **func**  
  IO$_DEACCESS

- **p1**  
  Not used.

- **p2**  
  Starting virtual address of quadword descriptor of the NCB. This
  parameter is used only if fields for diagnostic codes or local facilities
  are specified.

  For X.25 calls which use a PVC, do not specify p2, because none of the
  NCB fields are valid for a PVC.

- **p3**  
  Not used.

- **p4**  
  Not used.

- **p5**  
  Not used.

- **p6**  
  Unit number of the NV device.

**NCB Contents:**

Only mandatory, optional, and ignore items are listed in the following table. Other items will generate an error if you use them.

<table>
<thead>
<tr>
<th><strong>PSISC_NCB Item Code</strong></th>
<th><strong>Meaning</strong></th>
<th><strong>Notes</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>CAUSE</td>
<td>Code for PSDN clearing a call</td>
<td>1</td>
</tr>
<tr>
<td>DIAGCODE</td>
<td>Diagnostic code</td>
<td></td>
</tr>
<tr>
<td>LOCFAC</td>
<td>Local PSDN facilities</td>
<td></td>
</tr>
<tr>
<td>USERDATA</td>
<td>User data field</td>
<td>2</td>
</tr>
</tbody>
</table>

**Notes**

1. This field is ignored unless X.25 for OpenVMS is operating as a DCE (Data Circuit–terminating Equipment) to connect to other DTEs outside the PSDN. X.25 for OpenVMS can operate as a DCE, when using the ISO 8208 profile, to connect back–to–back with another DTE. DTEs cannot use nonzero codes, but DCEs can.

2. Use this field only for fast select calls. Up to 128 bytes of user data can be specified.
Examples

X.25 Code Example  In this example the system service QIO is used with function code IO$\_DEACCESS to clear the virtual circuit. The channel to the network device is PSI\_CHAN.

; Declaring the data:

PSI\_CHAN: .BLKW 1 ; Mailbox channel
IO\_STATUS: .BLKW 4 ; I/O status block
CLEAR\_NCB\_DESC: .LONG CLR\_NCB\_LEN ; NCB length
ADDRESS CLR\_NCB ; NCB address

; Using the System Service:

$QIOW\_S - ; Issue QIO and wait
CHAN = PSI\_CHAN,- ; to the network device
FUNC = #IO$\_DEACCESS,- ; Function is clear call
IOSB = IO\_STATUS,- ; I/O status block
P2 = #CLEAR\_NCB\_DESC ; NCB descriptor address

X.29 Code Example  In this example the system service QIO is used with function code IO$\_DEACCESS to clear the virtual circuit. The channel to the network device is PSI\_CHAN, and the NV device is NV\_UNIT.

NW\_CHAN: .BLKW 1 ; Channel to NW
NV\_UNIT: .BLKL 1 ; NV unit number
IO\_STATUS: .BLKW 4 ; I/O status block

$QIOW\_S - ; QIO and wait
IOSB = IO\_STATUS,- ; I/O status block
CHAN = NW\_CHAN,- ; channel to NW
FUNC = #IO$\_DEACCESS,- ; function is clear call
P6 = NV\_UNIT ; on NV terminal
BSBW IO\_ERROR ; Check system service and IOSB
Return Status

SS$_NORMAL  
Service successfully completed, the SVC has been cleared, or the PVC has been closed.

SS$_ACCVIO  
Unable to read argument p2.

SS$_FILALRACC  
Invalid unit number for SVC, or PVC already in use by another process.

SS$_IVBUFLEN  
The format of the NCB item-list is invalid. See the secondary status in the IOSB.

SS$_IVDEVNAM  
The format of the NCB is invalid, or an error was detected while processing the NCB. See the secondary status in the IOSB.

SS$_OPINCOMPL  
A previous call is still in progress on this channel.

Secondary Status Values

The secondary status values are found in the third word of the IOSB.

If the first word of the IOSB contains SS$_NORMAL, the third word can contain one or more of the following:

- PSI$M_STS_LOCDTELNG  
  Local DTE address too long—address truncated.

- PSI$M_STS_PKTBAD  
  Invalid packet size specified—nearest valid packet size chosen.

- PSI$M_STS_RPOALNG  
  RPOA item is not a multiple of 4 digits—truncated.

- PSI$M_STS_THRBAD  
  Invalid throughput class specified—nearest valid throughput class chosen.

- PSI$M_STS_USERLNG  
  Too much user data supplied—data truncated.

- PSI$M_STS_WINBAD  
  Invalid window size specified—nearest valid window size chosen.

- PSI$M_STS_WORDBAD  
  A word facility (one of the transit delay facilities) reduced to 65,535.

If the first word of the IOSB contains SS$_ABORT or SS$_IVDEVNAM, the third word can contain one of the following:

- PSI$C_ERR_BADNAME  
  Bad counted string parameter. This is probably due to a user-program error.

- PSI$C_ERR_BADPARM  
  Bad parameters specified.

- PSI$C_ERR_BAD_PVCNAME  
  Internal error, contact your local HP support representative for information about the variety of service options available to you and the procedures for submitting software problem reports.

- PSI$C_ERR_CONFLICT  
  Conflicting items specified.

- PSI$C_ERR_FACLNG  
  Facilities too long.

- PSI$C_ERR_INVEXP  
  Invalid use of expedited data negotiation.

- PSI$C_ERR_INVITEM  
  Invalid item code.

- PSI$C_ERR_INVNUM  
  Invalid ASCII number.
PSI$C_ERR_INVTRSTDLY
Invalid use of ISO end–to–end transit delay facility (PSI$C_NCB_MAX_TRST_DLY specified without PSI$C_NCB_ETE_TRST_DLY, or PSI$C_NCB_ETE_TRST_DLY specified without PSI$C_NCB_CUM_TRST_DLY).

PSI$C_ERR_L3ERR
Error returned from level 3.

PSI$C_ERR_NOLOCAL
The ACP has run out of local workspace memory. Increase the size of virtual memory.

PSI$C_ERR_NONONPAG
There is insufficient free nonpaged pool to complete the request. Increase the size of nonpaged pool and retry the request.

PSI$C_ERR_RECURLMNT
Recursion limit reached (probably internal error in data structures), contact your local HP support representative for information about the variety of service options available to you and the procedures for submitting software problem reports.

PSI$C_ERR_UNKNOWN
Unspecified internal error, contact your local HP support representative for information about the variety of service options available to you and the procedures for submitting software problem reports.

PSI$C_ERR_INVTRSTDLY
Invalid use of ISO end–to–end transit delay facility (PSI$C_NCB_MAX_TRST_DLY specified without PSI$C_NCB_ETE_TRST_DLY, or PSI$C_NCB_ETE_TRST_DLY specified without PSI$C_NCB_CUM_TRST_DLY).

If the first word of the IOSB contains any status value other than SS$_NORMAL, SS$_ABORT, or SS$_IVDEVNAM, the contents of the third word are meaningless.
2.12 $QIO(IO$_READVBLK) — Receive Data

$QIO(IO$_READVBLK) is used in programs to receive data transmitted from a remote DTE over the virtual circuit.

Format
$QIO [efn],chan,func,[iosb],[astadr],[astprm], p1,p2,[p3],[p4],[p5],[p6]

Arguments

<table>
<thead>
<tr>
<th>func</th>
<th>IO$_READVBLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Buffer address.</td>
</tr>
<tr>
<td>p2</td>
<td>Buffer length in bytes.</td>
</tr>
<tr>
<td>p3</td>
<td>Not used.</td>
</tr>
<tr>
<td>p4</td>
<td>Not used.</td>
</tr>
<tr>
<td>p5</td>
<td>Not used.</td>
</tr>
<tr>
<td>p6</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

Modifiers

IO$M_NOW Use this modifier to determine whether a message or part of a message has been received. The request with this modifier always completes immediately. If a message is available, the request completes with a status of SS$_NORMAL. If no message has been received, the request completes with a status of SS$_NODATA.

Example

In the following example, IO$_READVBLK needs to know the address and the size of the buffer where it places incoming data. The address of the buffer is READBUF and the size is READBUFSIZE. The channel to the network device is NW_CHAN.

; Declaring the data:
NW_CHAN:
    .BLKW 1 ; Network channel
IO_STATUS:
    .BLKW 4 ; I/O status block
READBUF:
    .BLKB 200 ; Buffer
    READBUFSIZE = .-READBUF

; Using the System Service:
$QIOM_S - ; Issue QIO and wait
    CHAN = NW_CHAN,- ; to NW
    FUNC = #IO$_READVBLK,- ; Function is read
    IOSB = IO_STATUS ; I/O status block
    P1 = READBUF,- ; Buffer address
    P2 = #READBUFSIZE ; and size
Return Status

SS$_NORMAL Service successfully completed, data received.
SS$_ACCVIO Cannot write to buffer described by arguments p1 and p2.
SS$_CLEARED The virtual circuit was cleared while this request was being processed, or the circuit was in the process of being cleared when you issued the request.
SS$_FILNOTACC A virtual circuit does not exist on this channel.
SS$_MEDOFL The PVC has been restarted. Issue a QIO system service call with a function code of IO$_NETCONTROL and a parameter of PSI$K_RESTART to confirm the restart and allow normal operation of the PVC.
SS$_NODATA No data has been received (IO$M_NOW only).
SS$_RESET The virtual circuit was reset while this request was being processed, or the circuit was in the process of being reset when you issued the request.

Secondary Status Values

The secondary status values are found in the third word of the IOSB.

If the first word of the IOSB contains SS$_NORMAL, the third word can contain one or more of the following:

PSI$M_QUALIFIED The data was a qualified message.
PSI$M_MOREDATA The operation (or request) completed before a packet (with the more data bit not set) was received. This may occur either because the buffer was not large enough to receive all the packets of a message, or because the request specified the IO$M_NOW modifier.

If the first word of the IOSB contains any other status value, the content of the third word is undefined.
2.13 $QIO(IO$_WRITEVBLK) — Transmit Data

$QIO(IO$_WRITEVBLK) is used in programs to transmit data over a virtual circuit.

Note that if your last QIO transmitted data with the IO$M_MORE qualifier, you can use $QIO(IO$_WRITEVBLK) with a zero data packet to send the remaining data from the previous QIO. Otherwise, if you send a zero data packet, $QIO(IO$_WRITEVBLK) will complete with a success status, but no data will be sent.

Format
$QIO [efn],chan,func,[iosb],[astadr],[astprm], p1,p2,[p3],[p4],[p5],[p6]

Arguments
func  IO$_WRITEVBLK
p1    Buffer address.
p2    Buffer length in bytes (maximum is 16,383).
p3    Not used.
p4    Not used.
p5    Not used.
p6    Not used.

Modifiers
IO$M_MORE Use this modifier if you want to associate the data of this IO$_ WRITEVBLK call with the data of the next IO$_WRITEVBLK call; that is, to indicate that more of the data message is to follow. Use this modifier to show that more of your data message is to follow with the next IO$_ WRITEVBLK call. See Note 1.

IO$M_QUALIFIED Use this modifier to distinguish a qualified message. Use qualified messages to specify a message which is different from usual; for example, a control message during a file transfer.

Notes
1. X.25 for OpenVMS sends packets of data when either of the following conditions is satisfied:
   • A packet is full.
   • You issue a QIO IO$_WRITEVBLK without the IO$M_MORE modifier. This indicates the end of a message to X.25 for OpenVMS.

For example, with a packet size of 128 bytes:
   • If you use the IO$_WRITEVBLK operation without the IO$M_MORE modifier to transmit a message with buffer length of 20 bytes, X.25 for OpenVMS sends a packet of 20 bytes.
   • If you use the IO$_WRITEVBLK operation without the IO$M_MORE modifier to transmit a message with buffer length of 200 bytes, X.25 for OpenVMS sends:
     - A packet of 128 bytes, which includes the more data bit
     - A packet of 72 bytes
• If you use the IO$_WRITEVBLK$ operation with the IO$M\_MORE$ modifier to transmit a message with buffer length of 200 bytes, X.25 for OpenVMS sends a packet of 128 bytes, and starts to fill the next packet.

• If you request three IO$_WRITEVBLK$ operations, each with a buffer length of 20 bytes and the IO$M\_MORE$ modifier, followed by an IO$_WRITEVBLK$ operation with a buffer length of 20 bytes and no IO$M\_MORE$ modifier, X.25 for OpenVMS sends one packet of 80 bytes.

Example
In the following example, IO$_WRITEVBLK$ needs the address and the length of the data to be transmitted. Here, the address is in Register 2 (R2) and the length in Register 3 (R3). The channel to the network device is PSI_CHAN.

; Declaring the data:
PSI_CHAN:
   .BLKW 1 ; Channel
IO_STATUS:
   .BLKW 4 ; I/O status block
; Using the System Service:
$QIOW\_S - ; Issue a QIO and wait
   CHAN = PSI_CHAN,- ; to the network device
   FUNC = #IO$_WRITEVBLK,- ; Function is write
   IOSB = IO_STATUS,- ; I/O status block
   P1 = (R2),- ; Buffer address
   P2 = R3 ; and size

Return Status
SS$_NORMAL$ Service successfully completed, buffer has been accepted for transmission.
SS$_ACCVIO$ The program does not have read access to the buffer, or cannot read all the buffer described by arguments p1 and p2.
SS$_CLEARED$ The virtual circuit was cleared while this request was being processed, or the circuit was in the process of being cleared when you issued the request.
SS$_FILNOTACC$ A virtual circuit does not exist on this channel.
SS$_MEDOFL$ The PVC has been restarted. Issue a QIO system service call with a function code of IO$_\_NETCONTROL$ and a parameter of PSI$K\_RESTART$ to confirm the restart and allow normal operation of the PVC.
SS$_RESET$ The virtual circuit was reset while this request was being processed, or the circuit was in the process of being reset when you issued the request.
Table 3–1 summarizes the system services specific to X.25 programming. These services are detailed in the remainder of this chapter.

<table>
<thead>
<tr>
<th>System Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_INTACK)</td>
<td>Confirms Receipt of an Interrupt</td>
</tr>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_INTERRUPT)</td>
<td>Transmits an Interrupt</td>
</tr>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_RESET)</td>
<td>Resets a Virtual Circuit or Confirms Receipt of a Reset</td>
</tr>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_RESTART)</td>
<td>Confirms Receipt of a Restart</td>
</tr>
</tbody>
</table>
3.1 $QIO(IO$_NETCONTROL, PSI$K_INTACK) — Confirm Receipt of an Interrupt

The QIO system service with a function code of IO$_NETCONTROL and a subfunction of PSI$K_INTACK confirms the receipt of an interrupt.

This service is only valid for X.25 calls. For X.29 calls, the NV device automatically acknowledges interrupts.

**Format**

$QIO [efn],chan,func,[iosb],[astadr],[astprm],[p1],[p2],[p3],[p4],[p5],[p6]

**Arguments**

- **func**  IO$_NETCONTROL
- **p1**     Not used.
- **p2**     Not used.
- **p3**     Not used.
- **p4**     Network control subfunction, PSI$K_INTACK, for confirming an interrupt.
- **p5**     Not used.
- **p6**     Unit number of the NV device. This must be zero in X.25 programs. The default value is zero.

**Example**

In the following example, the subfunction of the system service call confirms the receipt of an interrupt—the subfunction is expressed in **p4**. The channel to the network device is **INT_CHAN**.

```plaintext
; Declaring the data:
INT_CHAN:
   .BLKW 1 ; Network channel
IO_STATUS:
   .BLKW 4 ; I/O status block

; Using the System Service:
$QIO #S -
CHAN = INT_CHAN,- ; to the network
FUNC = #IO$_NETCONTROL,- ; Function is network control
IOSB = IO_STATUS,- ; I/O status block
P4 = #PSI$K_INTACK ; Subfunction is confirm interrupt received

Return Status

- **SS$_NORMAL** Service successfully completed, interrupt has been confirmed.
- **SS$_CLEARED** The virtual circuit was cleared while this request was being processed, or the circuit was in the process of being cleared when you issued the request.
- **SS$_FILNOTACC** A virtual circuit does not exist on this channel.
```
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS$_MEDOFL</td>
<td>The PVC has been restarted. Issue a QIO system service call with a function code of IO$_NETCONTROL and a parameter of PSI$K_RESTART to confirm the restart and allow normal operation of the PVC.</td>
</tr>
<tr>
<td>SS$_NOSOLICIT</td>
<td>No interrupt message has been received.</td>
</tr>
<tr>
<td>SS$_RESET</td>
<td>The virtual circuit was reset while this request was being processed, or the circuit was in the process of being reset when you issued the request.</td>
</tr>
</tbody>
</table>
### 3.2 $QIO(IO\_NETCONTROL, PSI\_K\_INTERRUPT) — Transmit an Interrupt

The QIO system service with a function code of IO\_NETCONTROL and a subfunction of PSI\_K\_INTERRUPT sends an interrupt over the virtual circuit.

Only one interrupt in each direction can be in progress over a virtual circuit at any time. Once you have sent an interrupt, it is not possible to send another until the remote DTE confirms receipt of the first. This is indicated by completion of the IO\_NETCONTROL operation that you used to send the first interrupt.

Note that a single interrupt may be in progress in each direction. Thus, you may receive an interrupt, and possibly confirm receipt of this incoming interrupt while awaiting confirmation of an outgoing one.

This means that cooperating processes could deadlock if they do not allow asynchronous code to perform an acknowledgment whilst waiting for a synchronous IO\_NETCONTROL operation to complete.

This service is only valid for X.25 calls. In X.29 calls, the NV device automatically sends interrupts as required by the X.29 protocol.

**Format**

$QIO [efn],chan,func,[iosb],[astadr],[astprm], p1,p2,[p3],p4,[p5],[p6]

**Arguments**

<table>
<thead>
<tr>
<th>func</th>
<th>IO_NETCONTROL</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Buffer address.</td>
</tr>
<tr>
<td>p2</td>
<td>Buffer length in bytes. Interrupts can be up to 32 octets long (depending on the PSDN), though most PSDNs restrict interrupts to one byte.</td>
</tr>
<tr>
<td>p3</td>
<td>Not used.</td>
</tr>
<tr>
<td>p4</td>
<td>Network control subfunction, PSI_K_INTERRUPT, for transmitting an interrupt.</td>
</tr>
<tr>
<td>p5</td>
<td>Not used.</td>
</tr>
<tr>
<td>p6</td>
<td>Not used.</td>
</tr>
</tbody>
</table>
Example
In the following example, the subfunction of the system service call transmits an interrupt—the subfunction is expressed in $p4$. The channel to the network device is INT_CHAN. The interrupt buffer is found at address INTBUF and its size is INTBUFSIZ.

; Declaring the data:
INT_CHAN:
  .BLKW 1 ; Network channel
IO_STATUS:
  .BLKW 4 ; I/O status block
INTBUF:
  .BLKB 10 ; Interrupt buffer
  INTBUFSIZ = .-INTBUF

; Using the System Service:
$QIOW_S -
CHAN = INT_CHAN,-
FUNC = #IO$_NETCONTROL-
IOSB = IO_STATUS,-
P1 = INTBUF,-
P2 = #INTBUFSIZ
P4 = #PSI$K_INTERRUPT ; Subfunction is interrupt

Return Status
SS$_NORMAL Service successfully completed, interrupt accepted for transmission.
SS$_ACCVIO The program does not have read access to the buffer.
SS$_CLEARED The virtual circuit was cleared while this request was being processed, or the circuit was in the process of being cleared when you issued the request.
SS$_DATAOVERUN More user data than allowed by PSDN. If the PSI$C_NCB_EXPEDITE item has been used during call set up with a value of 0, this error is returned for all values of $p2$ (Refer to Appendix C).
SS$_FILNOTACC A virtual circuit does not exist for this channel.
SS$_MEDOFL The PVC has been restarted. Issue a QIO system service call with a function code of IO$_NETCONTROL and a parameter of PSI$K_RESTART to confirm the restart and allow normal operation of the PVC.
SS$_OPINCOMPL A previous transmit interrupt request is still in progress.
SS$_RESET The virtual circuit was reset while this request was being processed, or the circuit was in the process of being reset when you issued the request.
3.3 $QIO(IO$_NETCONTROL, PSI$K_RESET) — Reset a Virtual Circuit or Confirm the Receipt of a Reset

The QIO system service with a function code of IO$_NETCONTROL and a subfunction of PSI$K_RESET resets a virtual circuit or confirms the receipt of a reset. The service resets a virtual circuit if no reset is outstanding, or confirms the receipt of a reset if one is outstanding. All pending messages are discarded if the virtual circuit is reset.

A return status of SS$_NORMAL does not guarantee that the remote DTE receives the diagnostic code. For example, the remote DTE may not receive the diagnostic code if a collision of resets occurs within the PSDN.

This service is only used for X.25 calls. In X.29 calls, the NV device automatically handles resets.

**Format**

$QIO [efn],chan.func,[iosb],[astadr],[astprm], [p1],[p2], [p3],p4,[p5],[p6]

**Arguments**

*func*  
IO$_NETCONTROL  

*p1*  
Not used.

*p2*  
Not used.

*p3*  
Diagnostic and cause code in a reset operation, or not used in a reset confirm operation.

The diagnostic code is specified in the low–order byte, parameter p3, and the cause code in the next byte. The cause code is ignored unless the ISO8208 profile is being used.

*p4*  
Network control subfunction, PSI$K_RESET, for resetting or confirming the reset of a virtual circuit.

*p5*  
Not used.

*p6*  
Not used.

**Example**

Here, the subfunction of the system service call transmits a reset or confirms reception of a reset—the subfunction is expressed in p4. The channel to the network device is NW_CHAN. Because p3 is not specified, the diagnostic code defaults to 0.

; Declaring the data:
NW_CHAN:  .BLKW 1  ; Network channel
IO_STATUS:  .BLKW 4  ; I/O status block

; Using the System Service:
$QIOW_S - ; Issue QIO and wait
CHAN = NW_CHAN,- ; to NW
FUNC = #IO$_NETCONTROL- ; Function is network control
IOSB = IO_STATUS,- ; I/O status block
P4 = #PSI$K_RESET ; Subfunction is reset
Return Status

SS$_NORMAL Service successfully queued (R0). This status indicates that the X.25 for OpenVMS software has reset the virtual circuit (and the reset has been confirmed), or confirms the reset of the virtual circuit (IOSB).

SS$_CLEARED The virtual circuit was cleared while this request was being processed, or the circuit was in the process of being cleared when you issued the request.

SS$_FILNOTACC The virtual circuit does not exist on this channel.

SS$_MEDOFL The PVC has been restarted. Issue a QIO system service call with a function code of IO$_NETCONTROL and a parameter of PSISK_RESTART to confirm the restart and allow normal operation of the PVC.

SS$_OPINCOMPL A previous reset request is still in progress.
3.4 $QIO(IO$_NETCONTROL, PSI$K_RESTART) — Confirm Receipt of a Restart

The QIO system service with a function code of IO$_NETCONTROL and a subfunction of PSI$K_RESTART confirms the receipt of a restart on a PVC.

**Format**

$QIO 
[efn],chan,func,[iosb],[astadr],|astprm], [p1],[p2],[p3],p4,[p5],[p6]

**Arguments**

*func* IO$_NETCONTROL

*p1* Not used.

*p2* Not used.

*p3* Not used.

*p4* Network control subfunction, PSI$K_RESTART, for confirming the restart of a virtual circuit.

*p5* Not used.

*p6* Not used.

**Example**

The subfunction of the system service call confirms the receipt of a restart. The subfunction is named in *p4*. The channel to the network device is PVC_CHAN.

```
; Declaring the data:
PVC_CHAN:
   .BLKW 1       ; Network channel
IO_STATUS:
   .BLKW 4       ; I/O status block
; Using the System Service:
$QIOW_S -     ; Issue QIO and wait
CHAN = PVC_CHAN,- ; to the circuit
FUNC = #IO$_NETCONTROL,- ; Function is network
       ; control
IOSB = IO_STATUS,- ; I/O status block
P4 = #PSI$K_RESTART ; Subfunction is restart
```

**Return Status**

<table>
<thead>
<tr>
<th>SS$_NORMAL</th>
<th>Service successfully completed, restart has been confirmed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS$_FILNOTACC</td>
<td>A virtual circuit does not exist for this channel.</td>
</tr>
<tr>
<td>SS$_NOSOLICIT</td>
<td>No restart has been received.</td>
</tr>
<tr>
<td>SS$_RESET</td>
<td>The virtual circuit was reset while this request was being processed, or the circuit was in the process of being reset when you issued the request.</td>
</tr>
</tbody>
</table>
Table 4–1 summarizes the system services specific to X.29 programming. These services are detailed in the remainder of this chapter.

For X.29 programming, functions supported by the terminal driver are available at the QIO interface. For details of QIO functions to the NV device, refer to the OpenVMS device driver documentation.

### Table 4–1 System Services Specific to X.29 Programming

<table>
<thead>
<tr>
<th>System Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$QIO(IO$_NETCONTROL, PSI$K_X29_READ)</td>
<td>Reads X.29 Terminal Characteristics</td>
</tr>
</tbody>
</table>

**READ Subfunctions:**

- PSI$K_X29_BREAK_ACTION: Returns the NV Action Descriptor Block
- PSI$K_X29_HANGUP_PARAMS: Reads the hangup PAD parameter template into a user–specified buffer
- PSI$K_X29_HOLD_TIMER: Returns the value of the Hold Timer
- PSI$K_X29_HOST_ECHO_PARAMS: Reads the host–echo PAD parameter template into a user–specified buffer
- PSI$K_X29_INT_ACTION: Returns the NV Action Descriptor Block
- PSI$K_X29_LOCAL_ECHO_PARAMS: Reads the local–echo PAD parameter template into a user–specified buffer
- PSI$K_X29_PAD_PARAMS: Returns the PAD Parameter List into the specified buffer
- PSI$K_X29_TEMP_NOHANG: Returns the setting of the TEMP_NOHANG flag.

(continued on next page)
### Table 4–1 (Cont.) System Services Specific to X.29 Programming

<table>
<thead>
<tr>
<th>System Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$QIO(IO_NETCONTROL, PSI_K_X29_READ_SPECIFIC)</td>
<td>Reads Specific X.29 Parameters</td>
</tr>
<tr>
<td>READ_SPECIFIC Subfunction:</td>
<td></td>
</tr>
<tr>
<td>PSI_K_X29_PAD_PARAMS</td>
<td>Returns specific PAD parameters into the specified buffer</td>
</tr>
<tr>
<td>$QIO(IO_NETCONTROL, PSI_K_X29_SET)</td>
<td>Sets X.29 Terminal Characteristics</td>
</tr>
<tr>
<td>SET Subfunctions:</td>
<td></td>
</tr>
<tr>
<td>PSI_K_X29_BREAK_ACTION</td>
<td>Defines the action, or series of actions, that NV is to take when an “Indication–of–Break” X.29 message is received</td>
</tr>
<tr>
<td>PSI_K_X29_HANGUP_PARAMS</td>
<td>Sets the hangup PAD parameter template</td>
</tr>
<tr>
<td>PSI_K_X29_HOST_ECHO_PARAMS</td>
<td>Sets the host–echo PAD parameter template</td>
</tr>
<tr>
<td>PSI_K_X29_HOLD_TIMER</td>
<td>Sets the Hold Timer to units of 1/10 of a second</td>
</tr>
<tr>
<td>PSI_K_X29_INT_ACTION</td>
<td>Sets the actions to be taken when an interrupt issued at the X.29 terminal is received as an X.25 Interrupt</td>
</tr>
<tr>
<td>PSI_K_X29_LOCAL_ECHO_PARAMS</td>
<td>Sets the local–echo PAD parameter template</td>
</tr>
<tr>
<td>PSI_K_X29_PAD_PARAMS</td>
<td>Assigns values to specified PAD parameters</td>
</tr>
<tr>
<td>PSI_K_X29_PAD_RESELECTION</td>
<td>Sends a PAD reselection message to the remote PAD</td>
</tr>
<tr>
<td>PSI_K_X29_TEMP_NOHANG</td>
<td>Sets the X.29 terminal to enable TEMP_NOHANG</td>
</tr>
</tbody>
</table>
4.1 $QIO(IO$_NETCONTROL, PSI$K_X29_READ) — Read X.29 Terminal Characteristics

$QIO(IO$_NETCONTROL, PSI$K_X29_READ) is used in X.29 programs to read PAD parameters and NV terminal characteristics.

This QIO performs an operation equivalent to the DCL command SHOW TERMINAL/X29.

Format
$QIO [efn],chan,func,[iosb],[astadr],[astprm], p1,p2,p3,p4,[p5],p6

Arguments

\(func\)  \hspace{1cm} IO$_NETCONTROL$

\(p1\)  \hspace{1cm} Buffer address.

\(p2\)  \hspace{1cm} Buffer size in bytes. The length of the buffer depends on the parameter \(p3\).

\(p3\)  \hspace{1cm} One of the following Network Control Subfunctions:

- PSI$K_X29_BREAK_ACTION
- PSI$K_X29_HANGUP_PARAMS
- PSI$K_X29_HOLD_TIMER
- PSI$K_X29_HOST_ECHO_PARAMS
- PSI$K_X29_INT_ACTION
- PSI$K_X29_LOCAL_ECHO_PARAMS
- PSI$K_X29_PAD_PARAMS
- PSI$K_X29_TEMP_NOHANG

These subfunctions are described in subsequent sections.

\(p4\)  \hspace{1cm} PSI$K_X29_READ$

\(p5\)  \hspace{1cm} Not used.

\(p6\)  \hspace{1cm} Unit number of the NV device.
4.1.1 READ Subfunction: PSI$K_X29_BREAK_ACTION

PSI$K_X29_BREAK_ACTION returns the NV Action Descriptor Block. The block contains details of the actions the NV device will take on receiving an Indication–of–break from the X.29 terminal.

The buffer size should be PSI$K_X29_ACTION_LENGTH (head = 20 bytes) to ensure the buffer is allocated sufficient space to hold the NV Action Descriptor Block that is returned.

Refer to the HP X.25 for OpenVMS—Programming Guide for a description of the NV Action Descriptor Block.

Example

NW_CHAN:
  .BLKW 1 ; Channel to NW
NV_UNIT:
  .BLKL 1 ; NV unit number
IO_STATUS:
  .BLKW 4 ; I/O Status block
DESC_BLOCK:
  .BLKB PSI$K_X29_ACTION_LENGTH ; Action Descriptor block
$QIOW_S - ; Issue QIO and wait
  IOSB = IO_STATUS,- ; I/O Status block
  CHAN = NW_CHAN,- ; Control channel to NW
  FUNC = #IO$_NETCONTROL,- ; Function is NETCONTROL
  P1 = DESC_BLOCK,- ; address of descriptor block
  P2 = #PSI$K_X29_ACTION_LENGTH,- ; length of descriptor block
  P3 = #PSI$K_X29_BREAK_ACTION,- ; Subfunction
  P4 = #PSI$K_X29_READ,- ; read data
  P6 = NV_UNIT ; NV unit number
BSBW IO_ERROR ; check system service and IOSB

Return Status

SS$_NORMAL Service completed successfully. Break actions are returned in buffer.

SS$_BUFFEROVF Buffer specified in P1 is too small to hold the data returned.
4.1.2 READ Subfunction: PSI$K_X29_HANGUP_PARAMS

This subfunction reads the hangup PAD parameter template into a user–specified buffer.

The hangup PAD parameter template defines PAD characteristics after an X.29 call has been cleared.

A buffer size of 256 bytes is normally adequate. This allows for 32 parameters.

Refer to the *HP X.25 for OpenVMS—Utilities Guide* for a description of PAD parameter templates.

**Example**

NW_CHAN:
  .BLKW 1 ; Channel to NW
IO_STATUS: ; I/O status block
  .BLKW 4

TEMPLATE_ENTRIES = 32 ; Allow for 32 parameters
TEMPLATE_BUFFER: ; Buffer to manipulate
  .BLKB TEMPLATE_ENTRIES * PSI$K_X29_PARAM_LENGTH
TEMPLATE_BUFFER_LEN = .-TEMPLATE_BUFFER ; template

; Read the hangup PAD parameter template
;
;-
$QIOW_S - ; Issue QIO and wait
  CHAN = NW_CHAN,- ; to network device
  FUNC = #IO$_NETCONTROL,- ; function is network control
  IOSB = IO_STATUS,- ; I/O status block
  P1 = TEMPLATE_BUFFER,- ; address of buffer
  P2 = #TEMPLATE_BUFFER_LEN,- ; length of buffer
  P3 = #PSI$K_X29_HANGUP_PARAMS,-
  - ; read subfunction specifies
    - ; which template to manipulate
  P4 = #PSI$K_X29_READ,- ; NV read operation
  P5 = NV_UNIT ; NV unit number
  BSBW IO_ERROR ; Check for I/O error
;+

**Return Status**

SS$.NORMAL Service completed successfully.
SS$_BUFFEROVF Buffer specified in P1 is too small to hold the template.
4.1.3 READ Subfunction: PSI$K_X29_HOLD_TIMER

This parameter returns the value of the Hold Timer into the specified buffer.

The timer value is in units of 1/10 of a second.

The buffer size should be set to 4 bytes to ensure the buffer is allocated sufficient space to hold the timer value returned.

**Example**

```
NW_CHAN:
    .BLKW 1 ; Channel to NW
NV_UNIT:
    .BLKL 1 ; NV unit number
IO_STATUS:
    .BLKW 4 ; I/O status block
TIMER:
    .BLKB 4 ; Hold Timer buffer
$QIOW_S - ; Issue QIO and wait
IOSB = IO_STATUS,- ; I/O Status Block
CHAN = NW_CHAN,- ; Control channel to NW
FUNC = #IOS_NETCONTROL,- ; Function is NETCONTROL
P1 = TIMER ; Address of output
    ; buffer
P2 = #4 ; Length of output
    ; buffer
P3 = #PSI$K_HOLD_TIMER,- ; Subfunction
P4 = #PSI$K_X29_READ,- ; Read data
P6 = NV_UNIT ; NV Unit number
BSBW IO_ERROR ; Check system service
    ; and IOSB
```

**Return Status**

- **SS$NORMAL**  
  Service completed successfully. Hold Timer value is returned.

- **SS$BUFFEROVF**  
  Buffer specified in p1 is too small to hold the value returned.
4.1.4 READ Subfunction: PSI$K_X29_HOST_ECHO_PARAMS

This subfunction reads the host–echo PAD parameter template into a user–specified buffer.

The host–echo PAD parameter template defines characteristics of the PAD in host–echo mode.

A buffer size of 256 bytes is normally adequate. This allows for 32 PAD parameters.

PAD parameter templates are described in the *HP X.25 for OpenVMS—Utilities Guide*.

Example

NW_CHAN:
  .BLKW 1 ; Channel to NW
NV_UNIT:
  .BLKL 1 ; NV unit number
IO_STATUS: ; I/O status block
  .BLQ 1

TEMPLATE_ENTRIES = 32 ; Allow for 32 parameters

TEMPLATE_BUFFER: ; Buffer to manipulate
  .BLKB TEMPLATE_ENTRIES * PSI$K_X29_PARAM_LENGTH

TEMPLATE_BUFFER_LEN = -. TEMPLATE_BUFFER ; template

; Read the Host-echo PAD parameter template
;
$QIOW_S - ; Issue QIO and wait
  CHAN = NW_CHAN,- ; to network device
  FUNC = #IOS_NETCONTROL,- ; function is network control
  IOSB = IO_STATUS,- ; I/O status block
  P1 = TEMPLATE_BUFFER,- ; address of buffer
  P2 = #TEMPLATE_BUFFER_LEN,- ; length of buffer
  P3 = #PSI$K_X29_HOST_ECHO_PARAMS,-
  - ; read subfunction specifies
  - ; which template to manipulate
  P4 = #PSI$K_X29_READ,- ; NV read operation
  P6 = NV_UNIT ; NV unit number

BSBW IO_ERROR ; Check for I/O error

Return Status

SS$_NORMAL Service completed successfully.
SS$_BUFFEROVF Buffer specified in p1 is too small to hold the template.
4.1.5 READ Subfunction: PSI$K_X29_INT_ACTION

This parameter returns the NV Action Descriptor Block. This contains details of the actions the NV device will take on receiving an Interrupt message from the X.29 terminal.

The buffer size should be PSI$K_X29_ACTION_LENGTH (= 20 bytes) to ensure the buffer is allocated sufficient space to hold the NV Action Descriptor Block that is returned.

For a description of the NV Action Descriptor Block, refer to the HP X.25 for OpenVMS—Programming Guide.

Example

```
NW_CHAN:
  .BLKW 1  ; Channel to NW
NV_UNIT:
  .BLKL 1  ; NV unit number
IO_STATUS:
  .BLKW 4  ; I/O Status block
DESC_BLOCK:
  .BLKB PSI$K_X29_ACTION_LENGTH  ; Action Descriptor block
$QIOW_S -  ; Issue QIO and wait
  IOSB = IO_STATUS,-  ; I/O Status block
  CHAN = NW_CHAN,-  ; Control channel to NW
  FUNC = #IO$NETCONTROL,-  ; Function is NETCONTROL
  P1 = DESC_BLOCK,-  ; address of descriptor block
  P2 = #PSI$K_X29_ACTION_LENGTH,-  ; length of descriptor block
  P3 = #PSI$K_X29_INT_ACTION,-  ; Subfunction
  P4 = #PSI$K_X29_READ,-  ; read data
  P6 = NV_UNIT  ; NV unit number
BSBW IO_ERROR  ; check system service and IOSB
```

Return Status

- **SS$NORMAL**  Service completed successfully. Interrupt actions are returned in buffer.
- **SS$BUFFEROVF**  Buffer specified in p1 is too small to hold the descriptor returned.
4.1.6 READ Subfunction: PSI$K_X29_LOCAL_ECHO_PARAMS

This subfunction reads the local–echo PAD parameter template into a user–specified buffer.

The local–echo PAD parameter template defines characteristics of the PAD in local–echo mode.

A buffer size of 256 bytes is normally adequate. This allows for 32 PAD parameters.

PAD parameter templates are described in the *HP X.25 for OpenVMS—Utilities Guide*.

Example

```
NW_CHAN:
  .BLKW 1 ; Channel to NW
NV_UNIT:
  .BLKL 1 ; NV unit number
IO_STATUS: ; I/O status block
  .BLKQ 1

TEMPLATE_ENTRIES = 32 ; Allow for 32 parameters
TEMPLATE_BUFFER: ; Buffer to manipulate
  .BLKB TEMPLATE_ENTRIES * PSI$K_X29_PARAM_LENGTH
TEMPLATE_BUFFER_LEN = .-TEMPLATE_BUFFER ; template

; Read the Local-echo PAD parameter template
;
-$QIOW_S - ; Issue QIO and wait
  CHAN = NW_CHAN,- ; to network device
  FUNC = #IOS$_NETCONTROL,- ; function is network control
  IOSB = IO_STATUS,- ; I/O status block
  P1 = TEMPLATE_BUFFER,- ; address of buffer
  P2 = #TEMPLATE_BUFFER_LEN,- ; length of buffer
  P3 = #PSI$K_X29_LOCAL_ECHO_PARAMS,-
      ; read subfunction specifies
  P4 = #PSI$K_X29_READ,- ; NV read operation
  P6 = NV_UNIT ; NV unit number
  BSBW IO_ERROR ; Check for I/O error
;+
```

Return Status

**SS$_NORMAL**  
Service completed successfully.

**SS$_BUFFEROVF**  
Buffer specified in *P1* is too small to hold the template.
4.1.7 READ Subfunction: PSI$K_X29_PAD_PARAMS

This subfunction returns the PAD Parameter List into the specified buffer.

The PAD Parameter List contains details of PAD parameters read. The actual length of the list is returned in the IOSB.

A buffer size of 512 is normally adequate. This allows for 64 PAD parameters.

Refer to the *HP X.25 for OpenVMS—Programming Guide* for a description of the PAD Parameter List. Refer to Appendix E for a complete description of the PAD parameters.

The following example and Figure 4–1 show:

- How the NV device (at the host DTE) communicates with a PAD.
- The contents of Register 0 (R0).
- The contents of the I/O Status Block (IOSB).

**Example**

```
NW_CHAN:
  .BLKW 1 ; Channel to NW
NV_UNIT:
  .BLKL 1 ; NV unit number
IO_STATUS:
  .BLKW 4 ; I/O Status block
PARAM_ENTRIES = 64 ; Allow for 64 parameters
PARAM_BUFFER: ; Buffer to manipulate parameters
  .BLKB PARAM_ENTRIES * PSI$K_X29_PARAM_LENGTH
PARAM_BUFFER_LEN = .-PARAM_BUFFER
$QIOW_S - ; Issue QIO and wait
IOSB = IO_STATUS,- ; I/O Status block
CHAN = NW_CHAN,- ; Control channel to NW
FUNC = #IO$_NETCONTROL,- ; Function is NETCONTROL
P1 = PARAM_BUFFER,- ; address of Parameter List
P2 = PARAM_BUFFER_LEN,- ; length of Parameter List
P3 = #PSI$K_X29_PAD_PARAMS ; Subfunction
P4 = #PSI$K_X29_READ,- ; read data
P6 = NV_UNIT ; NV unit number
BSBW IO_ERROR ; check system service and IOSB
```
Figure 4–1  $QIO(IO$_NETCONTROL!PSI$K_X29_READ) Operations

<table>
<thead>
<tr>
<th>System service call and status returns</th>
<th>NV device</th>
<th>PAD</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$QIO(IO$_NETCONTROL) p3=PSI$K_X29_PAD_PARAMS p4=PSI$K_X29_READ R0 status: SS$_NORMAL</td>
<td>READ PAD ALL</td>
<td></td>
<td>Requests all PAD parameters.</td>
</tr>
<tr>
<td>IOSB status: SS$_NORMAL</td>
<td></td>
<td></td>
<td>Possible outcomes:</td>
</tr>
<tr>
<td>IOSB status: SS$_TIMEOUT</td>
<td>PARAMETER INDICATION</td>
<td></td>
<td>1. Returns all PAD parameter. The IOSB returns the actual length of the data block.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. The PAD fails to return a message before the NV device times out.</td>
</tr>
</tbody>
</table>

**Return Status**

- **SS$\_NORMAL**  
  Service completed successfully: PAD parameters are returned in buffer supplied.
- **SS$\_BUFFEROVF**  
  Buffer specified in p1 is too small to hold the descriptor returned.
- **SS$\_CTRLERR**  
  A PAD error message is received in response to a message sent by NV.
- **SS$\_TIMEOUT**  
  No response received from PAD.
4.1.8 READ Subfunction: PSI$K_X29_TEMP_NOHANG

This returns the setting of the TEMP_NOHANG flag into the buffer supplied.

Only bit 0 is used. If PSI$K_X29_TEMP_NOHANG is set, the virtual circuit will not be cleared on a subsequent $DASSGN call.

The buffer size should be 4 to ensure the buffer is allocated sufficient space to hold the value returned.

Example

NW_CHAN:
  .BLKW 1 ; Channel to NW
TEMP_NOHANG_ON:
  .BLKL 1 ; Value of temp nohang
NV_UNIT:
  .BLKL 1 ; NV unit number
IO_STATUS:
  .BLKW 4 ; I/O Status block

;+ Read the temp_nohang bit
;-
$QIOW_S - ; QIO and wait
IOSB = IO_STATUS,- ; I/O status block
CHAN = NW_CHAN,- ; channel to NW
FUNC = #IOS_NETCONTROL,- ; function is net control
P1 = TEMP_NOHANG_ON,- ; output buffer
P2 = #4,- ; longword
P3 = #PSI$K_X29_TEMP_NOHANG,- ; subfunction
P4 = #PSI$K_X29_READ,- ; NV read operation
P6 = NV_UNIT ; NV unit number
BSBW IO_ERROR ; Check system service and IOSB

Return Status

SS$NORMAL Service completed successfully: TEMP_NOHANG flag is returned in the buffer supplied.

SS$BUFFEROVF The buffer specified in P1 is too small to return the data.
4.2 $QIO(IO$_NETCONTROL, PSI$K_X29_READ_SPECIFIC) —
Read Specific X.29 Parameters

$QIO(IO$_NETCONTROL, PSI$K_X29_READ_SPECIFIC) is used in X.29 programs to read specific PAD parameters.

**Format**
$QIO [efn],[chan,func,[iosb],[astadr],[astprm]}, p1,p2,p3,p4,[p5],p6

**Arguments**

<table>
<thead>
<tr>
<th>func</th>
<th>IO$_NETCONTROL</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Buffer address.</td>
</tr>
<tr>
<td>p2</td>
<td>Buffer size in bytes. The length of the buffer depends on the parameter p3.</td>
</tr>
<tr>
<td>p3</td>
<td>The following Network Control Subfunction:</td>
</tr>
<tr>
<td></td>
<td>PSI$K_X29_PAD_PARAMS</td>
</tr>
<tr>
<td></td>
<td>The subfunction is described in the following section.</td>
</tr>
<tr>
<td>p4</td>
<td>PSI$K_X29_READ_SPECIFIC</td>
</tr>
<tr>
<td>p5</td>
<td>Not used.</td>
</tr>
<tr>
<td>p6</td>
<td>Unit number of the NV device.</td>
</tr>
</tbody>
</table>
4.2.1 READ_SPECIFIC Subfunction: PSI$K_X29_PAD_PARAMS

This subfunction returns specific PAD parameters into the specified buffer.

The buffer described by $p1$ and $p2$ is read to get a list of the parameters that should be returned. NV issues an X.29 read parameters message to the PAD, requesting the parameters listed in the buffer.

On completion of the QIO, the buffer is overwritten with the parameters returned from the PAD.

For a description of the PAD Parameter List, refer to the HP X.25 for OpenVMS—Programming Guide. For a complete description of the PAD parameters, refer to Appendix E.

Figure 4–2 shows:
- How the NV driver (at the host DTE) communicates with a PAD.
- The contents of Register 0 (R0).
- The contents of the I/O Status Block (IOSB).

Figure 4–2 $QIO(IO\_NETCONTROL!PSI$K_X29_READ_SPECIFIC) Operations

<table>
<thead>
<tr>
<th>System service call and status returns</th>
<th>NV device</th>
<th>PAD</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$QIO(IO_NETCONTROL) p3=PSI$K_X29_PAD_PARAMS p4=PSI$K_X29_READ_SPECIFIC</td>
<td>READ PAD</td>
<td></td>
<td>Requests specific PAD parameters.</td>
</tr>
<tr>
<td>R0 status: SS$_NORM$</td>
<td></td>
<td></td>
<td>Possible outcomes:</td>
</tr>
<tr>
<td>IOSB status: SS$_NORM$</td>
<td></td>
<td>PARAMETER INDICATION</td>
<td>1. Returns all PAD parameter. The IOSB returns the actual length of the data block.</td>
</tr>
<tr>
<td>IOSB status: SS$_TIMEOUT$</td>
<td></td>
<td></td>
<td>2. The PAD fails to return a message before the NV device times out.</td>
</tr>
</tbody>
</table>
Example
In this example, QIO(IO$_NETCONTROL) is used to read the parameters specified in PARAM_BUFFER (ECHO and network–specific parameter 42).

```
.MACRO PAD_PARAM_ITEM CODE, VALUE=0, ATTR=0
  .WORD CODE ; PSI$W_X29_PARAM_REF
  .WORD ATTR ; PSI$W_X29_PARAM_FLAGS
  .BYTE VALUE ; PSI$B_X29_PARAM_VALUE
  .BYTE 0, 0, 0 ; Must be zero
.ENDM

NW_CHAN: .BLKW 1 ; Channel to NW
NV_UNIT: .BLKL 1 ; NV unit number
IO_STATUS: ; I/O status block
  .BLKW 4

PARAM_BUFFER:
ECHO: PAD_PARAM_ITEM ; find out the echo setting
  CODE=PSI$K_X29_PAR_ECHO
PAD_PARAM_ITEM ; Network specific parameters follow
  CODE=0, VALUE=0
PAR42: PAD_PARAM_ITEM ; find network specific parameter 42
  CODE=42
PARAM_BUFFER_LEN = .-PARAM_BUFFER ;

; Get the PAD parameters
;-
$QIOW_S - ; Issue QIO and wait
  CHAN = NW_CHAN,- ; to network device
  FUNC = #IO$_NETCONTROL,- ; Function is network control
  IOSB = IO_STATUS,- ; I/O status block
  P1 = PARAM_BUFFER,- ; Address of buffer
  P2 = PARAM_BUFFER_LEN,- ; Length of buffer
  P3 = #PSI$K_X29_PAD_PARAMS,- ; Subfunction specifies
  - ; read PAD parameters
  P4 = #PSI$K_X29_READ_SPECIFIC,- ; NV read specific operation
  P6 = NV_UNIT ; NV unit number
BSBW IO_ERROR ; Check for I/O error
```

Return Status

SS$_NORMAL Service completed successfully: PAD parameters are returned in buffer supplied.

SS$_CTRLERR A PAD error message is received in response to a message sent by NV.

SS$_TIMEOUT No response received from PAD.
4.3 $QIO(IO$_NETCONTROL, PSI$K_X29_SET) — Set X.29 Terminal Characteristics

$QIO(IO$_NETCONTROL, PSI$K_X29_SET) is used in X.29 programs to set PAD parameters and NV terminal characteristics.

This QIO performs an operation equivalent to the DCL command SET TERMINAL/X29.

Figure 4–3 shows:
- How the NV device (at the host DTE) communicates with a PAD.
- The contents of Register 0 (R0).
- The contents of the I/O Status Block (IOSB).

### Figure 4–3 $QIO(IO$_NETCONTROL!PSI$K_X29_SET) Operations

<table>
<thead>
<tr>
<th>System service call and status returns</th>
<th>NV device</th>
<th>PAD</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$QIO(IO$_NETCONTROL) p3=PSI$K_X29_PAD_PARAMS p4=PSI$K_X29_SET R0 status: SS$_NORMAL</td>
<td>Message contains the PAD parameters to be set.</td>
<td>SET AND READ PAD</td>
<td>Possible outcomes:</td>
</tr>
<tr>
<td>IOSB status: SS$_NORMAL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOSB status: SS$_BADPARAM</td>
<td>PARAMETER INDICATION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOSB status: SS$_TIMEOUT</td>
<td>PARAMETER INDICATION OF ERRORS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Returns all PAD parameter. The IOSB returns the actual length of the data block.
2. The IOSB returns the number of the first parameter in error, and its value.
3. The PAD fails to return a message before the NV device times out.
Format
$QIO \ [efn],\ chan,\ func,\ [iosb],\ [astadr],\ [astprm],\ p1,p2,p3,p4,p5,p6$

Arguments

\[\]
\[\]

$func$ \ IO\-_NETCONTROL

$p1$ \ Buffer address.

$p2$ \ Buffer size in bytes. The length of the buffer depends on the parameter $p3$.

$p3$ \ One of the following Network Control Subfunctions:

\begin{itemize}
  \item PSI$\_X29\_BREAK\_ACTION$
  \item PSI$\_X29\_HANGUP\_PARAMS$
  \item PSI$\_X29\_HOLD\_TIMER$
  \item PSI$\_X29\_HOST\_ECHO\_PARAMS$
  \item PSI$\_X29\_INT\_ACTION$
  \item PSI$\_X29\_LOCAL\_ECHO\_PARAMS$
  \item PSI$\_X29\_PAD\_PARAMS$
  \item PSI$\_X29\_PAD\_RESELECTION$
  \item PSI$\_X29\_TEMP\_NOHANG$
\end{itemize}

These subfunctions are described in subsequent sections.

$p4$ \ PSI$\_X29\_SET$

$p5$ \ Not used.

$p6$ \ Unit number of the NV device.
4.3.1 SET Subfunction: PSI\$K_X29_BREAK_ACTION

This parameter defines the action, or series of actions, that NV is to take when an Indication–of–break X.29 message is received.

Usually, PSI\$K_X29_BREAK_ACTION is specified for when you issue a BREAK command while using the host–based PAD.

Parameter \( p2 \) specifies the size of the NV Action Descriptor Block. Set \( p2 \) either to PSI\$K_X29_ACTION_LENGTH ( = 20 bytes), or to a value between 4 and 20 bytes.

Set one or more of the following bits in the 4–byte Action flag (byte 0 to 3):

- PSI\$V_X29_ACTION_RESET To reset the circuit.
- PSI\$V_X29_ACTION_PURGE To purge all input in the NV device.
- PSI\$V_X29_ACTION_CLEAR To clear the call.

Only the first three bits of byte 0 are used.

If you require other actions to be taken, set the counted string PSI\$T_X29_ACTION_STRING in the NV Action Descriptor Block.

For details of the NV Action Descriptor Block, refer to the *HP X.25 for OpenVMS—Programming Guide*.

**Example**

```assembly
NW_CHAN:
   .BLKW 1 ; Channel to NW
NV_UNIT:
   .BLKL 1 ; NV Unit number
IO_STATUS:
   .BLKW 4 ; I/O Status block
DESC_BLOCK:
   .LONG PSI\$M_X29_ACTION_PURGE!- ;Descriptor block
   PSI\$M_X29_ACTION_RESET
STRING:
   .BYTE STRING_LENGTH-1 ;Count
   .BYTE 24 ;^X
   .BYTE 15 ;^0
STRING_LENGTH = .-STRING
DESC_LENGTH = .-DESC_BLOCK
; $QIOW_S - ; Issue QIO and wait
IOSB = IO_STATUS ; I/O Status block
CHAN = NW_CHAN,- ;Control channel to NW
FUNC = #IOS$_NETCONTROL,- ;Function is NETCONTROL
P1 = DESC_BLOCK,- ;Address of descriptor block
P2 = #DESC_LENGTH,- ;Length of descriptor block
P3 = #PSI\$K_X29_BREAK_ACTION,- ;Subfunction
P4 = #PSI\$K_X29_SET,- ;NV set data
P6 = NV_UNIT ;NV Unit number
; BSBW IO_ERROR ;Check system service and IOSB
```

4–18 X.29 System Services
<table>
<thead>
<tr>
<th>Return Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS$_NORMAL</td>
<td>Service completed successfully: break accepted for transmission.</td>
</tr>
<tr>
<td>SS$_ACCVIO</td>
<td>The NV Action Descriptor Block specified in p1 cannot be read.</td>
</tr>
</tbody>
</table>
4.3.2 SET Subfunction: PSI$K_X29_HANGUP_PARAMS

This subfunction sets the hangup PAD parameter template.

The hangup PAD parameter template defines the PAD characteristics after an X.29 call has been cleared.

Parameter p2 is the size of the PAD parameter buffer. Calculate p2 as follows:

\[
(number \text{ of } PAD \text{ parameters to be set}) \times PSI$K_X29\_PARAM\_LENGTH
\]

Refer to the *HP X.25 for OpenVMS—Utilities Guide* for a description of how to use PAD parameter templates.

**Example**

This example sets the hangup PAD parameter template to turn on echo and editing.

```assembly
.MACRO PAD_PARAM_ITEM CODE, VALUE=0, ATTR=0
  .WORD CODE ; PSI$W_X29\_PARAM\_REF
  .WORD ATTR ; PSI$W_X29\_PARAM\_FLAGS
  .BYTE VALUE ; PSI$B_X29\_PARAM\_VALUE
  .BYTE 0, 0, 0 ; Must be zero
 .ENDM

NW_CHAN:
  .BLKW 1 ; Channel to NW
IO_STATUS: ; I/O status block
  .BLKQ 1

TEMPLATE_BUFFER:
  PAD_PARAM_ITEM - ; Turn on Echo
    CODE=PSI$K_X29\_PAR\_ECHO, VALUE=1
  PAD_PARAM_ITEM - ; Turn on Editing
    CODE=PSI$K_X29\_PAR\_EDIT, VALUE=1
  TEMPLATE_BUFFER_LEN = .-TEMPLATE_BUFFER ; template

; Set the PAD parameter template
;$QIOW_S - ; Issue QIO and wait
  CHAN = NW_CHAN,- ; to network device
  FUNC = #IOS$\_NETCONTROL,- ; Function is net control
  IOSB = IO\_STATUS,- ; I/O status block
  P1 = TEMPLATE\_BUFFER,- ; Address of buffer
  P2 = TEMPLATE\_BUFFER\_LEN,- ; Length of buffer
  P3 = #PSI$K_X29\_HANGUP\_PARAMS,-
    - ; Subfunction specifies
    - ; which template to manipulate
  P4 = #PSI$K_X29\_SET,- ; NV set operation
  P5 = NV\_UNIT ; NV unit number
  BSBW IO\_ERROR ; Check for I/O error
```

**Return Status**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS$_NORMAL</td>
<td>Service completed successfully.</td>
</tr>
<tr>
<td>SS$_ACCVIO</td>
<td>The buffer specified in p1 cannot be read.</td>
</tr>
</tbody>
</table>
4.3.3 SET Subfunction: PSI$K_X29_HOST_ECHO_PARAMS

This subfunction sets the host–echo PAD parameter template.

The host–echo PAD parameter template defines characteristics of the PAD in host–echo mode.

Parameter p2 is the size of the PAD parameter buffer. Calculate p2 as follows:

\[(\text{number of PAD parameters to be set}) \times \text{PSI$K_X29_PARAM_LENGTH}\]

Refer to the *HP X.25 for OpenVMS—Utilities Guide* for a description of how to use PAD parameter templates.

If the NV device is in host–echo mode, the NV device will consult the new template, and configure the PAD according to the instructions in the template.

Example

This example sets the host–echo PAD parameter template so that echo and editing are turned off, the user’s timeout value is used, and wrap and newline are turned off.

```assembly
.MACRO PAD_PARAM_ITEM CODE, VALUE=0, ATTR=0
.WORD CODE ; PSI$W_X29_PARAM_REF
.WORD ATTR ; PSI$W_X29_PARAM_FLAGS
.BYTE VALUE ; PSI$B_X29_PARAM_VALUE
.BYTE 0, 0, 0 ; Must be zero
.ENDM

NW_CHAN: .BLKW 1 ; Channel to NW
IO_STATUS: ; I/O status block
.NV_UNIT: .BLKL 1 ; NV unit number

TEMPLATE_BUFFER: PAD_PARAM_ITEM - ; Turn Echo off
    CODE=PSI$K_X29_PAR_ECHO, VALUE=0
PAD_PARAM_ITEM - ; Turn Editing off
    CODE=PSI$K_X29_PAR_EDIT, VALUE=0
PAD_PARAM_ITEM - ; Use the user's timeout value
    CODE=PSI$K_X29_PAR_TIMEOUT, ATTR=PSI$M_X29_USER_VALUE
PAD_PARAM_ITEM - ; Turn off wrap
    CODE=PSI$K_X29_PAR_WRAP, VALUE=0
PAD_PARAM_ITEM - ; Turn off Newline
    CODE=PSI$K_X29_PAR_NEW_LINE, VALUE=0

TEMPLATE_BUFFER_LEN = .-TEMPLATE_BUFFER ; template

; Set the PAD parameter template
-$QIOW_S - ; Issue QIO and wait
CHAN = NW_CHAN,- ; to network device
FUNC = #IO$_NETCONTROL,- ; Function is net control
IOSB = IO_STATUS,- ; I/O status block
P1 = TEMPLATE_BUFFER,- ; Address of buffer
P2 = TEMPLATE_BUFFER_LEN,- ; Length of buffer
P3 = #PSI$K_X29_HOST_ECHO_PARAMS,-
    ; Subfunction specifies
    ; which template to manipulate
P4 = #PSI$K_X29_SET,- ; NV set operation
P6 = NV_UNIT ; NV unit number
BSBW IO_ERROR ; Check for I/O error
```
<table>
<thead>
<tr>
<th>Return Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS$_NORMAL</td>
<td>Service completed successfully.</td>
</tr>
<tr>
<td>SS$_ACCVIO</td>
<td>The buffer specified in $pI$ cannot be read.</td>
</tr>
</tbody>
</table>
4.3.4 SET Subfunction: PSI$K_X29_HOLD_TIMER

This parameter sets the Hold Timer to units of 1/10 of a second.

Parameter \( p2 \) is the size of the buffer that stores the timer value. Set \( p2 \) to 4 bytes.

Set the timer to 0 to transmit all output from NV to the X.29 terminal immediately.

For details of how the Hold Timer functions in relation to the NV device, refer to the HP X.25 for OpenVMS—Programming Guide.

Example

In this example the Hold Timer is set to 1/5 second.

```assembly
NW_CHAN:
  .BLKW 1 ; Channel to NW
NV_UNIT:
  .BLKL 1 ; NV Unit number
IO_STATUS:
  .BLKW 4 ; I/O Status block
HOLD_TIMER:
  .LONG 2 ; Hold timer value

$QIOW_S - ; Issue QIO and wait
IOSB = IO_STATUS ; I/O Status block
CHAN = NW_CHAN,- ; Control channel to NW
FUNC = #IOS$NETCONTROL,- ; Function is NETCONTROL
P1 = HOLD_TIMER ; New hold timer
P2 = #4 ; Longword
P3 = #PSI$K_X29_HOLD_TIMER,- ; Subfunction
P4 = #PSI$K_X29_SET,- ; NV set data
P6 = NV_UNIT ; NV Unit number

; BSBW IO_ERROR ; Check system service and IOSB
```

Return Status

- **SS$_NORMAL** Service completed successfully: Hold Timer is set.
- **SS$_ACCVIO** The timer value specified in the \( pI \) buffer cannot be read.
4.3.5 SET Subfunction: PSI$K_X29_INT_ACTION

This parameter sets the actions to be taken when an interrupt issued at the X.29 terminal is received as an X.25 Interrupt.

Parameter $p2$ is the size of the NV Action Descriptor Block. Set $p2$ either to PSI$K_X29_ACTION_LENGTH (= 20 bytes), or to a value between 4 and 20 bytes.

Set one or more of the following bits in the 4-byte Action flag (byte 0 to 3):

PSI$V_X29_ACTION_RESET To reset the virtual circuit.
PSI$V_X29_ACTION_PURGE To purge all input in the NV device.
PSI$V_X29_ACTION_CLEAR To clear the call.

Only the first three bits of byte 0 are used.

If you require other actions to be taken, set the counted string (PSI$T_X29_ACTION_STRING) in the NV Action Descriptor Block.

For details of the NV Action Descriptor Block, refer to the HP X.25 for OpenVMS—Programming Guide.

Example

In the following example, the Typeahead buffer is purged automatically and the NV input is purged by setting PSI$V_X29_ACTION_PURGE in the Action flag and Ctrl/Y in the counted string.

```
NW_CHAN:
  .BLKW 1 ; Channel to NW
NV_UNIT:
  .BLKL 1 ; NV Unit number
IO_STATUS:
  .BLKW 4 ; I/O Status block
DESC_BLOCK: ;Descriptor block
  .LONG PSI$M_X29_ACTION_PURGE ;Purge NV
;
STRING:
  .BYTE STRING_LENGTH ;Count
  .BYTE 25 ;^Y
STRING_LENGTH = .-STRING
DESC_LENGTH = .-DESC_BLOCK
;
$QIOW_S - ; Issue QIO and wait
IOSB = IO_STATUS ; I/O Status block
CHAN = NW_CHAN,- ;Control channel to NW
FUNC = #IOS_NETCONTROL,- ;Function is NETCONTROL
P1 = DESC_BLOCK,- ;Address of descriptor block
P2 = #DESC_LENGTH,- ;Length of descriptor block
P3 = #PSI$K_X29_INT_ACTION,- ;Subfunction
P4 = #PSI$K_X29_SET,- ;NV set data
P6 = NV_UNIT ;NV Unit number
;
BSBW IO_ERROR ;Check system service and IOSB
```

Return Status

SS$NORMAL Service completed successfully: interrupt actions are set.
SS$ACCVIO The NV Action Descriptor Block specified in $p1$ cannot be read.
4.3.6 SET Subfunction: PSI$K_X29_LOCAL_ECHO_PARAMS

This subfunction sets the local–echo PAD parameter template.

The local–echo PAD parameter template defines characteristics of the PAD in local–echo mode.

Parameter $p2$ is the size of the PAD parameter buffer. Calculate $p2$ as follows:

\[
(p\text{number of PAD parameters to be set}) \times \text{PSI}K\_X29\_PARAM\_LENGTH
\]

Refer to the *HP X.25 for OpenVMS—Utilities Guide* for a description of how to use PAD parameter templates.

**Example**

In the following example the local–echo PAD parameter template is set to calculate echo, turn editing on, turn off timeouts and set up the newline parameter.

```plaintext
.MACRO PAD_PARAM_ITEM CODE, VALUE=0, ATTR=0
.WORD CODE ; PSI$W_X29_PARAM_REF
.WORD ATTR ; PSI$W_X29_PARAM_FLAGS
.BYTE VALUE ; PSI$B_X29_PARAM_VALUE
.BYTE 0, 0, 0 ; Must be zero
.ENDM

NW_CHAN: .BLKW 1 ; Channel to NW
NV_UNIT: .BLKL 1 ; NV unit number
IO_STATUS: ; I/O status block .BLKW 4

TEMPLATE_BUFFER: PAD_PARAM_ITEM - ; Calculate echo
                        CODE=PSI$K_X29_PAR_ECHO, ATTR=PSI$M_X29_CALCULATE
PAD_PARAM_ITEM - ; Turn Editing on
                        CODE=PSI$K_X29_PAR_EDIT, VALUE=1
PAD_PARAM_ITEM - ; Turn off timeouts
                        CODE=PSI$K_X29_PAR_TIMEOUT, VALUE=0
PAD_PARAM_ITEM - ; Set up newline
                        CODE=PSI$K_X29_PAR_NEW_LINE, VALUE=4

TEMPLATE_BUFFER_LEN = .-TEMPLATE_BUFFER ; template

; Set the PAD parameter template

$QIOM_S - ; Issue QIO and wait
            CHAN = NW_CHAN,- ; to network device
            FUNC = #IO$_NETCONTROL,- ; Function is network control
            IOSB = IO_STATUS,- ; I/O status block
            P1 = TEMPLATE_BUFFER,- ; Address of buffer
            P2 = TEMPLATE_BUFFER_LEN,- ; Length of buffer
            P3 = #PSI$K_X29_LOCAL_ECHO_PARAMS,-
                - ; Subfunction specifies
                - ; which template to manipulate
            P4 = #PSI$K_X29_SET,- ; NV set operation
            P5 = NV_UNIT ; NV unit number
            BSBW IO_ERROR ; Check for I/O error
```
<table>
<thead>
<tr>
<th>Return Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS$NORMAL</td>
<td>Service completed successfully.</td>
</tr>
<tr>
<td>SS$ACCVIO</td>
<td>The buffer specified in $pI$ cannot be read.</td>
</tr>
</tbody>
</table>
4.3.7 SET Subfunction: PSI$K_X29_PAD_PARAMS

This parameter assigns values to specified PAD parameters.

Parameter $p2$ is the size of the PAD Parameter List. Calculate $p2$ as follows:

\[
(number \ of \ PAD \ parameters \ to \ be \ set) \times PSI$K_X29_PARAM_LENGTH
\]

The PAD Parameter List consists of PAD parameter items. To set standard PAD parameters, you should specify the appropriate Parameter Code in the type field, followed by the required value.

Refer to Appendix E for details of the PAD parameters, and refer to the HP X.25 for OpenVMS—Programming Guide for details of the PAD Parameter List and how PAD parameter settings may affect NV operation.

To set nonstandard PAD parameters, specify a sequence of items in the PAD Parameter List, as follows:

- Item 1: Parameter code = 0  
  Parameter value = 0
- Item 2: Nonstandard PAD parameter code  
  Parameter value
- Item 3: Nonstandard PAD parameter code  
  Parameter value

... and so on.

For details of the nonstandard PAD facilities that are supported, refer to the technical documentation supplied by the PSDN.

Example

The following code turns echo off by setting ECHO to zero.

```
.MACRO PAD_PARAM_ITEM CODE, VALUE=0, ATTR=0
.WORD CODE ; PSI$W_X29_PARAM_REF
.WORD ATTR ; PSI$W_X29_PARAM_FLAGS
.BYTE VALUE ; PSI$B_X29_PARAM_VALUE
.BYTE 0, 0, 0 ; Must be zero
.ENDM

NW_CHAN:
.BLKW 1 ; Channel to NW
NV_UNIT:
.BLKL 1 ; NV Unit number
IO_STATUS:
.BLKW 4 ; I/O status block
PAD_PARAM_BLOCK:
  PAD_PARAM_ITEM - ; Turn Echo off
  CODE=PSI$K_X29_PAR_ECHO, VALUE=0
  PAD_PARAM_LEN = .-PAD_PARAM_BLOCK
  ;+ 
  ; Make sure that the PAD echo parameter is turned off
;-
  $QIOW_S - ; QIO and wait
  CHAN = NW_CHAN,- ; NW channel
  IOSB = IO_STATUS,- ; I/O status block
  FUNC = #IOS_NETCONTROL,- ; network control operation
  P1 = PAD_PARAM_BLOCK,- ; PAD Parameter List
  P2 = #PAD_PARAM_LEN,- ; length of block
  P3 = #PSI$K_X29_PAD_PARAMS,- ; change PAD parameter
  P4 = #PSI$K_X29_SET,- ; NV set operation
  P6 = NV_UNIT ; NV unit number
  BSBW IO_ERROR ; Check system service and IOSB
```
**Return Status**

- **SS$_NORMAL**: Service completed successfully: PAD parameters are set.
- **SS$_ACCVIO**: The PAD Parameter List specified in pI cannot be written.
- **SS$_BADPARAM**: Unable to set PAD parameters, incorrect specification. Refer to the description of secondary status values below for details.
- **SS$_BUFFEROVF**: Buffer specified in pI is too small to hold the PAD parameters that are set and are read back into that buffer.
- **SS$_CTRLERR**: A PAD error message is received in response to a message sent by NV. Refer to the description of secondary status below for details.
- **SS$_TIMEOUT**: No response received from PAD.

**Secondary Return Status**

Secondary status values are found in the third word of the IOSB.

If the first word of the IOSB contains **SS$_BADPARAM**, the third word will contain the PAD parameter number (IOSB word 4 contains the original value).

If the first word of the IOSB contains **SS$_CTRLERR**, contents of the third word will be:

- Byte 1: X.29 error type
- Byte 2: Invalid code
4.3.8 SET Subfunction: PSI$K_X29_PAD_RESELECTION

This subfunction sends a PAD reselection message to the remote PAD.

The NCB item–list contains the parameters to put in the reselection message, remote DTE address, user data field and facilities.

Parameter \( p1 \) is the address of the start of the NCB item–list.

Parameter \( p2 \) is the length of the NCB item–list.

NCB Contents:

Only mandatory, optional, and ignore items are listed in the following table. Other items will generate an error if you use them.

<table>
<thead>
<tr>
<th>PSISC_NCB Item Code</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Mandatory items</strong></td>
<td></td>
</tr>
<tr>
<td>RESELECT_DTE</td>
<td>Reselection DTE address</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Optional items</strong></td>
<td></td>
</tr>
<tr>
<td>LOCFAC</td>
<td>Local PSDN facilities</td>
<td></td>
</tr>
<tr>
<td>USERDATA</td>
<td>User data field</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td><strong>Ignored items</strong></td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>Null item identifier</td>
<td></td>
</tr>
</tbody>
</table>

Notes

1. The user data field can be up to 16 bytes in length for normal calls and up to 128 bytes in length for fast select calls.
Example

The following example sends a PAD reselection message to the remote PAD, with DTE address 234273400321, with optional user data and local PSDN facilities.

```
.MACRO NCB_ITEM_HEADER CODE, LENGTH=0
 .WORD 4+(LENGTH)
 .WORD CODE
 .ENDM

NW_CHAN:
 .BLKW 1 ; Channel to NW

IO_STATUS: ; I/O status block
 .BLKQ 1

RESELECT_ITEM_LIST:
 NCB_ITEM_HEADER psi$c_ncb_reselect_dte, 4+13
 .ascic '234273400321'
 NCB_ITEM_HEADER psi$c_ncb_user_data, 4+7
 .ascic 'BARRY'
 NCB_ITEM_HEADER psi$c_ncb_locfac, 4+7
 .byte ^x43, ^x07, ^x07 ; Window size of 7
 .byte ^x42, ^x0a, ^x0a ; Packet size of 1024

RESELECT_ITEM_LIST_LEN = .-RESELECT_ITEM_LIST
;
; Set the PAD parameter template
;
$QIOW_S - ; Issue QIO and wait
 CHAN = NW_CHAN,- ; to network device
 FUNC = #IO$_NETCONTROL,- ; Function is net control
 IOSB = IO_STATUS,- ; I/O status block
 P1 = RESELECT_ITEM_LIST,- ; Address of buffer
 P2 = RESELECT_ITEM_LIST_LEN,- ; Length of buffer
 P3 = #PSI$K_X29_PAD_RESELECTION,- ; Subfunction specifies
 - ; send pad reselection message
 P4 = #PSI$K_X29_SET,- ; NV set operation
 P6 = NV_UNIT ; NV unit number
 BSW = IO_ERROR ; Check for I/O error

Return Status

SS$_NORMAL The PAD reselection message has been sent to the PAD.
SS$_ACCVIO The buffer specified in p1 cannot be read.
SS$_IVBUFLEN The NCB is badly formatted; check the item lengths.
SS$_BADPARAM An item code is invalid or the contents of an item are invalid.
```
4.3.9 SET Subfunction: PSI$K_X29_TEMP_NOHANG

This parameter sets the X.29 terminal to enable TEMP_NOHANG.

Normally, when the last channel is deassigned from an NV unit, the call is cleared and all resources used by the unit are returned to the system. This subfunction temporarily disables terminal hangup and allows the NV unit to be passed to another process. Refer to the HP X.25 for OpenVMS—Programming Guide for further details.

When the channel has been deassigned and the login sequence has started, setting the TEMP_NOHANG flag to 0 does not affect operations.

Set $p3$ either to 1 (enable TEMP_NOHANG) or to 0 (disable TEMP_NOHANG). Set $p2$ (buffer length) to 4 bytes.

Example

The following code enables TEMP_NOHANG.

```assembly
NW_CHAN:
  .BLKW 1 ; Channel to NW
TEMP_NOHANG_ON:
  .LONG 1 ; Value to set in temp no hang
NV_UNIT:
  .BLKL 1 ; NV unit number
IO_STATUS:
  .BLKW 4 ; I/O status block

;+ ; Set the temp_nohang bit ;-
$QIOW_S - ; QIO and wait
IOSB = IO_STATUS,- ; I/O status block
CHAN = NW_CHAN,- ; channel to NW
FUNC = #IOS$NETCONTROL,- ; function is net control
P1 = TEMP_NOHANG_ON,- ; value to set
P2 = #4,- ; longword
P3 = #PSI$K_X29_TEMP_NOHANG,- ; change NV temp_nohang
P4 = #PSI$K_X29_SET,- ; NV set operation
P6 = NV_UNIT ; NV unit number
BSBW IO_ERROR ; Check system service and IOSB

Return Status

SS$_NORMAL Service completed successfully: TEMP_NOHANG is set.
SS$_ACCVIO The value in the $p1$ buffer cannot be written.
```
Status Codes Returned at System Service Completion

When a system service completes, a status code is returned. The system services used for X.25 for OpenVMS programming place the return status code in Register 0 (R0). Return status codes usually show if the service completed successfully, although sometimes they simply provide information for your program. Moreover, a success return status code (severity level = 1) does not necessarily mean that the program achieved the desired result, but only that the service has completed all its functions, and has returned control to the calling program. For example, the return status code SS$_BUFFEROVF$, returned when a character string returned by a service is longer than the buffer provided to receive it, is a success code.

Note

All of the QIO system service calls return a success status code of SS$_NORMAL$. This indicates only that the request was successfully queued.

Warning status codes (and some error status codes) show that the service may have completed part, but not all, of the requested functions.

Generally, return status codes have the same meaning wherever they are returned. The return status codes for each system service are listed in the following chapters, together with any special meanings of the status codes for that system service:

- Chapter 2 for system services common to X.25 and X.29 programming.
- Chapter 3 for system services specific to X.25 programming.
- Chapter 4 for system services specific to X.29 programming.

When your program calls a system service, read the descriptions of the service’s return status codes to determine whether you want the program to check for particular return conditions.

When a system service completes, the system services used for X.25 for OpenVMS programming place the return status code in the first word of the I/O Status Block (IOSB), in addition to placing the return status code in Register 0 (R0). Further I/O completion status information is placed in the second, third, and fourth words of the IOSB, as shown in Table 5–1.
Table 5–1 Completion Status Information in the IOSB

<table>
<thead>
<tr>
<th>IOSB</th>
<th>Contents</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1</td>
<td>Return status</td>
<td>The completion status code returned by the system service call.</td>
</tr>
<tr>
<td>Word 2</td>
<td>Byte count</td>
<td>The number of bytes that have been processed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For read operations: the number of bytes read.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For IO$_ACCESS$ operations: the number of bytes of the NCB processed successfully before an error.</td>
</tr>
<tr>
<td>Word 3</td>
<td>Secondary status</td>
<td>For IO$_ACCESS$ and IO$_DEACCESS$:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If IOSB word 1 contains SS$_ABORT$ or SS$_IVDEVNAM$, word 3 contains a secondary status. In addition, on OpenVMS VAX systems, if IOSB word 1 contains SS$_NORMAL$, word 3 contains a secondary status. Otherwise, the contents are undefined. Details of the secondary statuses for IO$_ACCESS$ and IO$_DEACCESS$ are provided with the service descriptions in Section 2.6 and Section 2.11.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For IO$_ACPCONTROL$:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If IOSB word 1 contains SS$_IVDEVNAM$, word 3 contains a secondary status. Otherwise, the contents are undefined. Details of the secondary status for IO$_ACPCONTROL$ are with the service description in Section 2.10.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For IO$_NETCONTROL$(PSI$K_X29_SET,PSI$K_X29_PAD_PARAMS):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If IOSB word 1 contains SS$_BADPARAM$ or SS$_CTRLERR$, word 3 contains a secondary status. Otherwise, the contents are undefined. Details of the secondary status for IO$_NETCONTROL$(PSI$K_X29_SET,PSI$K_X29_PAD_PARAMS) are with the service description in Section 4.3.7.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For IO$_READVBLK$:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If IOSB word 1 contains SS$_NORMAL$, word 3 contains a secondary status. Otherwise, the contents are undefined. Details of the secondary status for IO$_READVBLK$ are with the service description in Section 2.12.</td>
</tr>
<tr>
<td>Word 4</td>
<td>—</td>
<td>Internal information, ignore this field.</td>
</tr>
</tbody>
</table>

Refer to the OpenVMS documentation for further details of the use of asynchronous system traps (ASTs), I/O status blocks (IOSBs) and event flags.

The operating system does not automatically handle system service failure or warning conditions. You must test for them, and handle them yourself. This contrasts with the operating system’s handling of exception conditions that are detected by the hardware or software. The operating system handles these exception conditions by default. However, you can override the default handling.
by declaring a condition handler (refer to the OpenVMS documentation of system services).

5.1 Testing the Return Status Code

Each language provides a mechanism for testing the return status. Often, you need check only the low–order bit, such as testing for TRUE (success or informational return) or FALSE (error or warning return).

However, you can check the entire value for a specific return condition. To permit this, each language provides a way for your program to determine the values associated with specific, symbolically defined codes. Always use these symbolic names when your code tests for specific conditions.

For information on how to test for these symbolically defined codes, see the User's Guide for your programming language.

The return status is stored as a binary value in a longword. Depending on your specific needs, you can test just the low–order bit, the three low–order bits, or the entire value:

- The low–order bit indicates successful (1) or unsuccessful (0) completion of the service.
- The three low–order bits, taken together, represent the severity of the error. Severity code values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Severity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Warning</td>
</tr>
<tr>
<td>1</td>
<td>Success</td>
</tr>
<tr>
<td>2</td>
<td>Error</td>
</tr>
<tr>
<td>3</td>
<td>Informational</td>
</tr>
<tr>
<td>4</td>
<td>Severe (or fatal) error</td>
</tr>
<tr>
<td>5–7</td>
<td>(Reserved)</td>
</tr>
</tbody>
</table>

- The remaining bits (bits 3 to 31) classify the particular return condition and the operating system component that issued the status code. Note that for system service return status values, the high–order word (bits 16 through 31) contains zeros.

Each numeric status code has a symbolic name in the format:

`SS$_code`

where `code` is a mnemonic describing the return condition. For example, the most common successful return is indicated by `SS$_NORMAL`, and a common error status code is `SS$_ACCVIO` (access violation, indicating that the service could not read an input argument, or write an output argument).

The symbols associated with the different return status values are defined in the default system library.
5.2 Special Return Conditions

Two process execution modes affect the way control is returned to your program when an error occurs during the execution of a system service. These modes are:

- Resource wait mode
- System service failure exception mode

If you choose to change the default setting for either of these modes, your program must handle the special conditions that result.

5.2.1 Resource Wait Mode

Many system services require certain system resources for execution. These resources include system dynamic memory and process quotas for I/O operations. Normally, when a system service is called and a required resource is not available, the program is placed in a wait state, until the resource becomes available. The service then completes execution. This mode is called resource wait mode.

In a real–time environment, however, it may not be practical or desirable for a program to wait. You can choose to disable resource wait mode in such cases and control will return immediately to your program with an error status code. You can enable or disable resource wait mode with the Set Resource Wait Mode ($SETRWM) system service.

How a program responds to the lack of a resource depends on the application, and the particular service that is being called. In some instances, the program may want to continue execution and retry the service call later. In other instances, it may be necessary only to note that the program is being required to wait.

5.2.2 System Service Failure Exception Mode

This mode determines whether control is returned to you in the normal manner following an error in a system service operation, or whether an exception is generated. System service failure exception mode is disabled by default: your program receives control following an error. You can enable and disable system service failure exception mode with the Set System Service Failure Exception Mode ($SETSFM) service.

High–level language compilers generate calls to system services for many statements or instructions in source programs (for example, reads and writes to files generate calls to VAX RMS, which uses the QIO and QIOW services). If you enable system service failure exception mode, many different types of errors (such as an I/O attempt to a nonexistent device or non–numeric input to a mathematics routine) will generate the message:

%SYSTEM-F-SSFAIL, system service failure exception,…

Because of this, you are recommended not to enable system service failure exception mode in high–level language programs, except perhaps when debugging. If you enable system service failure exception mode and do not declare your own condition handler, many error messages displayed at run time will be meaningless.
5.3 Obtaining Values for Other Symbolic Codes

In addition to the symbolic codes for specific return conditions, many individual services also have symbolic codes for the offsets, identifiers, or flags associated with these services. For example, the Create Process ($CREPRC) service, which is used to create a subprocess or a detached process, has symbolic codes associated with the various privileges and quotas you can grant to the created process.

If your language has a method of obtaining values for these symbols, that method will be explained in the User's Guide for your programming language. If your language does not have such a method:

- Write a short MACRO program containing the desired macros.
- Assemble the program and generate a listing. Use the listing to find the desired symbols and their hexadecimal values.
- Define each symbol with its value within your source program.
Summary of X.25 System Service Calls

A.1 System Services for Setting Up and Clearing Communications

\$ASSIGN \text{devnam}, \text{chan}, [\text{acmode}], [\text{mbxnam}]

Assign a channel.

\$CANCEL \text{chan}

Clear a virtual call on a channel.

\$CREMBX [\text{prmflg}], \text{chan}, [\text{maxmsg}], [\text{bufquo}], [\text{promsk}], [\text{acmode}], [\text{lognam}]

Create mailbox and assign a channel.

\$QIO [\text{efn}], \text{chan}, \text{IO\$_ACCESS}, [\text{iosb}], [\text{astadr}], [\text{astprm}], [p1], [p2], [p3], [p4], [p5], [p6]

Set up a virtual circuit.

\$QIO [\text{efn}], \text{chan}, \text{IO\$_DEACCESS}, [\text{iosb}], [\text{astadr}], [\text{astprm}], [p1], [p2], [p3], [p4], [p5], [p6]

Clear a virtual circuit.

\$DASSGN \text{chan}

Deassign a channel.

A.2 System Services for Handling Incoming Calls

\$QIO [\text{efn}], \text{chan}, \text{IO\$_ACCESS}!\text{IO\$_M\_ACCEPT}, [\text{iosb}], [\text{astadr}], [\text{astprm}], [p1], [p2], - [p3], [p4], [p5], [p6]

Accept a request to set up a virtual circuit.

\$QIO [\text{efn}], \text{chan}, \text{IO\$_ACCESS}!\text{IO\$_M\_ABORT}, [\text{iosb}], [\text{astadr}], [\text{astprm}], [p1], [p2], - [p3], [p4], [p5], [p6]

Reject a request to set up a virtual circuit.

\$QIO [\text{efn}], \text{chan}, \text{IO\$_ACCESS}!\text{IO\$_M\_REDIRECT}, [\text{iosb}], [\text{astadr}], [\text{astprm}], [p1], [p2], - [p3], [p4], [p5], [p6]

Redirect a call request.

\$QIO [\text{efn}], \text{chan}, \text{IO\$_ACP\_CONTROL}, [\text{iosb}], [\text{astadr}], [\text{astprm}], [p1], [p2], [p3], [p4], [p5], [p6]

Declare a process as a network process.
A.3 System Services for Handling Control and Data Messages

$QIO \text{[efn}, \text{chan}, \text{IO$_\text{READVBLK}$}, [iosb], [astadr][astprm], p1,p2,[p3],[p4],[p5],[p6]$

Receive data.

$QIO \text{[efn}, \text{chan}, \text{IO$_\text{WRITEVBLK}$}, [iosb],[astadr][astprm], p1,p2,[p3],[p4],[p5],[p6]$

Transmit data.

$QIO \text{[efn}, \text{chan}, \text{IO$_\text{NETCONTROL}$}, [iosb],[astadr],[astprm],[p1],[p2],[p3], -$\text{PSI$_\text{K_INTACK}$}, [p5],[p6]$

Confirm receipt of an interrupt.

$QIO \text{[efn}, \text{chan}, \text{IO$_\text{NETCONTROL}$}, [iosb],[astadr],[astprm],[p1],[p2],[p3], -$\text{PSI$_\text{K_INTERRUPT}$}, [p5],[p6]$

Transmit an interrupt.

$QIO \text{[efn}, \text{chan}, \text{IO$_\text{NETCONTROL}$}, [iosb],[astadr],[astprm],[p1],[p2],[p3], -$\text{PSI$_\text{K_RESET}$}, [p5],[p6]$

Reset a virtual circuit, or confirm the receipt of a reset.

$QIO \text{[efn}, \text{chan}, \text{IO$_\text{NETCONTROL}$}, [iosb],[astadr],[astprm],[p1],[p2],[p3], -$\text{PSI$_\text{K_RESTART}$}, [p5],[p6]$

Confirm receipt of a restart.

$QIO \text{[efn}, \text{chan}, \text{IO$_\text{NETCONTROL}$}, [iosb],[astadr],[astprm],p1,p2,p3, -$\text{PSI$_\text{K_X29_READ}$}, [p5],[p6]$

Read X.29 terminal characteristics.
Summary of X.29 System Service Calls

B.1 System Services for Setting Up and Clearing Communication

$ASSIGN devnam, chan, [acmode], [mbxnam]
Assign a channel.

$GETDVI [efn], [chan], [devnam], itmlst, [iosb], [astadr], [astprm], nullarg
Get NV unit number.

$CREMBX [prmflg], chan, [maxmsg], [bufquo], [promsk], [acmode], [lognam]
Create mailbox and assign a channel.

$QIO [efn], chan, IO$_ACCESS, [iosb], [astadr], [astprm], [p1], [p2], [p3], [p4], [p5], [p6]
Set up a virtual circuit.

$QIO [efn], chan, IO$_DEACCESS, [iosb], [astadr], [astprm], [p1], [p2], [p3], [p4], [p5], [p6]
Clear a virtual circuit.

$DASSGN chan
Deassign a channel.

In the above $QIO system service calls, the arguments are as follows:

$p2$ is the start address of the quadword NCB descriptor.

$p6$ (where mandatory) is the unit number of the NV device.

B.2 System Services for Handling Incoming Calls

$QIO [efn], chan, IO$_ACPCONTROL, [iosb], [astadr], [astprm], [p1], [p2]
Declare a process as a network process.

$QIO [efn], chan, IO$_ACCESS|IO$_M_ACCEPT, [iosb], - [astadr], [astprm], [p1], [p2], [p3], [p4], [p5], [p6]
Accept a request to set up a virtual circuit.

$QIO [efn], chan, IO$_ACCESS|IO$_M_ABORT, [iosb], - [astadr], [astprm], [p1], [p2], [p3], [p4], [p5], [p6]
Reject a request to set up a virtual circuit.

$QIO [efn], chan, IO$_ACCESS|IO$_M_REDIRECT, [iosb], - [astadr], [astprm], [p1], [p2], [p3], [p4], [p5], [p6]
Redirect a call request.

In IO$_ACPCONTROL:

$p1$ is the address of the quadword descriptor of a block containing:

 BYTE NFB$C_DECLNAME
 .LONG 0
$p2$ is the address of the quadword descriptor of a Network Process Declaration Block.

In the other $QIO$ system service calls, the arguments are as follows:

$p2$ is the start address of the quadword NCB descriptor.

$p6$ (where mandatory) is the unit number of the NV device.

**B.3 System Services for Reading and Setting PAD Parameters and NV Terminal Characteristics**

$QIO$ 

Read X.29 terminal characteristics.

Set X.29 terminal characteristics.

For both these system services, the arguments are as follows:

$p1$ is the buffer address.

$p2$ is the buffer size (bytes).

$p3$ is one of the following Network Control Subfunctions:

- PSI$K_X29\_BREAK\_ACTION$
- PSI$K_X29\_HANGUP\_PARAMS$
- PSI$K_X29\_HOLD\_TIMER$
- PSI$K_X29\_HOST\_ECHO\_PARAMS$
- PSI$K_X29\_INT\_ACTION$
- PSI$K_X29\_LOCAL\_ECHO\_PARAMS$
- PSI$K_X29\_PAD\_PARAMS$
- PSI$K_X29\_PAD\_RESELECTION$ (PSI$K_X29\_SET$ only)
- PSI$K_X29\_TEMP\_NOHANG$

$p6$ is the unit number of the NV device.

**B.4 Terminal Driver Functions**

The functions supported by the terminal driver are available at the QIO interface. For details of the terminal driver QIOs, refer to the VMS terminal driver documentation.
C

Network Connect Block (NCB)

C.1 Description of the NCB

For each call request, X.25 for OpenVMS constructs a Network Connect Block (NCB). The NCB holds information about how the call is to be routed across the PSDN, charging and diagnostic information, and requests for transmission facilities. It can also contain some user data. Your program uses the NCB to access and amend this information when it sets up or clears a virtual circuit, or when it accepts, redirects, or rejects a request to set up a virtual circuit.

This appendix describes the contents and format of the NCB. The format of the NCB is described in Section C.2. The items that make up the NCB are listed by function in Section C.4 and defined in alphabetical order in Section C.5.

C.2 NCB Format

A Network Connect Block (NCB) consists of items of information. The items are of variable length, each item containing the following fields:

Word 1: Length of data in this item (including words 1 and 2).
Word 2: Type code, of the form PSI$C_NCB_code.
Subsequent bytes: Data (variable length).

Each type of data has a specific format: single-byte value, single word value, counted string, or longword value.

C.3 Data Type Format Definitions

The data type formats used in NCBs are:

- single byte: An 8-bit field
- single word: A 16-bit field
- longword: A 32-bit field
- counted string: A variable length field where the first octet contains the number of bytes in the remainder of the field
C.4 NCB Item Functions

This section describes the content and usage of each NCB item in alphabetical order. Table C–1 summarizes the items by type.

For an outgoing call request, the NCB must include some routing information. For an SVC, specify the remote DTE address. For a PVC, specify the PVC identifier. You may also need to specify the network identifier.

For an incoming call request, X.25 for OpenVMS constructs an NCB with an appropriate Incoming Call Identifier.

Other items in the NCB are either optional, ignored, or not used, as documented in the system service descriptions.

If you specify an item that is not used with a particular system service, X.25 for OpenVMS generates an error. In these cases, you may change the item code to null, so that you can re-use the NCB without having to delete the length and data fields for that item.

Your PSDN must subscribe to a facility for your program to be able to specify it.

<table>
<thead>
<tr>
<th>Code</th>
<th>Data (usage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUG</td>
<td>(Bilateral) Closed User Group</td>
</tr>
<tr>
<td>DTECLASS</td>
<td>Name of the DTE Class from which a member DTE is used to make the call</td>
</tr>
<tr>
<td>FILTER</td>
<td>Filter entity</td>
</tr>
<tr>
<td>FLT_PRI</td>
<td>Destination priority (incoming)</td>
</tr>
<tr>
<td>FLT_REDPRI</td>
<td>Redirection priority (incoming)</td>
</tr>
<tr>
<td>GATEWAY</td>
<td>Gateway identifier (reserved usage)</td>
</tr>
<tr>
<td>ICI</td>
<td>Incoming Call Identifier</td>
</tr>
<tr>
<td>LOC_DTE</td>
<td>Local DTE address</td>
</tr>
<tr>
<td>LOC_SUBADR</td>
<td>Local subaddress</td>
</tr>
<tr>
<td>REM_DTE</td>
<td>Remote DTE address—this must be included in the NCB for SVCs if a remote DTE address is not defined in the template used for the call</td>
</tr>
<tr>
<td>REM_SUBADR</td>
<td>Remote subaddress</td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>Name of template created by network management, with specified parameters</td>
</tr>
</tbody>
</table>

Routing information: PVC only

<table>
<thead>
<tr>
<th>Code</th>
<th>Data (usage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASSWORD</td>
<td>User password</td>
</tr>
<tr>
<td>PVC_NAM</td>
<td>PVC identifier—this must be included in the NCB for PVCs</td>
</tr>
</tbody>
</table>

(continued on next page)
Table C–1 (Cont.) NCB Item Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Data (usage)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Routing information: PSDN facilities</strong></td>
<td></td>
</tr>
<tr>
<td>ADDR_MOD_RSN</td>
<td>Reason for modifying called line address (incoming)</td>
</tr>
<tr>
<td>CALLED_EXTENSION</td>
<td>Called address extension</td>
</tr>
<tr>
<td>CALLING_EXTENSION</td>
<td>Calling address extension</td>
</tr>
<tr>
<td>CALL_REDIR.ORIG</td>
<td>Original DTE destination of redirected call (incoming)</td>
</tr>
<tr>
<td>CALL_REDIR_RSN</td>
<td>Call redirection reason (incoming)</td>
</tr>
<tr>
<td>LOCFAC</td>
<td>Local PSDN facilities (outgoing)</td>
</tr>
<tr>
<td>LOCFACR</td>
<td>Local PSDN facilities (incoming)</td>
</tr>
<tr>
<td>NET_USER_ID</td>
<td>Network user identifier</td>
</tr>
<tr>
<td>RPOA</td>
<td>Remote Port Of Access</td>
</tr>
<tr>
<td><strong>User data</strong></td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>Null item identifier</td>
</tr>
<tr>
<td>USERDATA</td>
<td>User data field</td>
</tr>
<tr>
<td><strong>User data: SVC only</strong></td>
<td></td>
</tr>
<tr>
<td>FSEL</td>
<td>Fast select without restriction (outgoing)</td>
</tr>
<tr>
<td>FSEL_RES</td>
<td>Fast select with restricted response (outgoing)</td>
</tr>
<tr>
<td>RESPDATA</td>
<td>Fast select response data (outgoing)</td>
</tr>
<tr>
<td><strong>Diagnostics</strong></td>
<td></td>
</tr>
<tr>
<td>CAUSE</td>
<td>Code for PSDN clearing a call</td>
</tr>
<tr>
<td>DIAGCODE</td>
<td>Diagnostic code</td>
</tr>
<tr>
<td>REASON</td>
<td>Code for X.25 for OpenVMS clearing a call</td>
</tr>
<tr>
<td><strong>Charging information</strong></td>
<td></td>
</tr>
<tr>
<td>REVCHG</td>
<td>Reverse charging request (outgoing)</td>
</tr>
<tr>
<td><strong>Charging information: PSDN facilities</strong></td>
<td></td>
</tr>
<tr>
<td>CHARGE_MON</td>
<td>Monetary units for charging (incoming)</td>
</tr>
<tr>
<td>CHARGE_SEG</td>
<td>Segment count for charging (incoming)</td>
</tr>
<tr>
<td>CHARGE_TIME</td>
<td>Elapse time for charging (incoming)</td>
</tr>
<tr>
<td>CHARGING_INFO</td>
<td>Charging information request (outgoing)</td>
</tr>
</tbody>
</table>

(continued on next page)
<table>
<thead>
<tr>
<th>Code</th>
<th>Data (usage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKTSIZE</td>
<td>Packet size (outgoing)</td>
</tr>
<tr>
<td>THRUCLS</td>
<td>Throughput class (maximum data rate)</td>
</tr>
<tr>
<td>RCV_QUOTA</td>
<td>Total size of receive buffers (in bytes)</td>
</tr>
<tr>
<td>WINSIZE</td>
<td>Window size (outgoing)</td>
</tr>
</tbody>
</table>

**Transmission parameters: PSDN facilities**

<table>
<thead>
<tr>
<th>Code</th>
<th>Data (usage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUM_TRST_DLY</td>
<td>Cumulative transit delay (outgoing)</td>
</tr>
<tr>
<td>CUM_TRST_DLY_R</td>
<td>Cumulative transit delay (incoming)</td>
</tr>
<tr>
<td>ETE_TRST_DLY</td>
<td>End–to–end transit delay (outgoing)</td>
</tr>
<tr>
<td>EXPEDITE</td>
<td>Negotiate use of expedited data (interrupts)</td>
</tr>
<tr>
<td>MAX_TRST_DLY</td>
<td>Maximum acceptable transit delay</td>
</tr>
<tr>
<td>MIN_THRUCLS</td>
<td>Minimum throughput class (for data rate)</td>
</tr>
<tr>
<td>TRANSIT_DELAY</td>
<td>Requested maximum transit delay</td>
</tr>
</tbody>
</table>
C.5 NCB Item Descriptions

PSI$C_NCB_ADDR_MOD_RSN: Reason for modifying called line address

The PSDN specified a code in this item to indicate why it has modified the address that was called. Your program can specify a PSDN–specific code in this item to indicate why it has accepted or cleared an incoming call.

Data format: single–byte value in the range 0–255.

PSI$C_NCB_CALLED_EXTENSION: Called address extension

This item specifies an address extension for the destination (See the description of the IO$ACCESS system service for more information).

Data format: counted string (second byte in the string contains a nibble count).

PSI$C_NCB_CALLING_EXTENSION: Calling address extension

This item specifies the address extension of the DTE that originated an incoming call. (See the description of the IO$ACCESS system service for more information).

Data format: counted string (second byte in the string contains a nibble count).

PSI$C_NCB_CALL_REDIR_ORIG: Original DTE destination of redirected call

This item specifies the DTE from which a call was redirected.

Data format: counted string.

PSI$C_NCB_CALL_REDIR_RSN: Call redirection reason

This item specifies why the PSDN redirected the call.

Data format: single–byte value in the range 0–255.

PSI$C_NCB_CAUSE: Code for PSDN clearing a call

Your PSDN uses this item to specify a PSDN–specific code indicating why it cleared a call.

If the ISO8208 profile is being used, your program may specify a value in the item when it clears a call. Values 1–127 are available if the interface is acting as a DCE.

Data format: single–byte value in the range 0–255.

PSI$C_NCB_CHARGE_MON: Monetary units for charging

This item specifies the charge for the call in monetary units. It may be supplied by the PSDN when the call is cleared.

Data format: counted string, PSDN–specific format.

PSI$C_NCB_CHARGE_SEG: Segment count for charging

This item specifies the charge for the call in segment counts. It may be supplied by the PSDN when the call is cleared.

Data format: counted string, PSDN–specific format.
PSI$C_NCB_CHARGE_TIME: Elapse time for charging
   This item specifies the charge for the call in elapsed time. It may be specified by the PSDN when the call is cleared.
   Data format: counted string, PSDN–specific format.

PSI$C_NCB_CHARGING_INFO: Charging information request
   Your program can use this item to request charging information when it sends a call request.
   Data format: no data field.

PSI$C_NCB_CUG: (Bilateral) Closed User Group
   This item specifies a (Bilateral) Closed User Group as destination.
   Data format: counted string.

PSI$C_NCB_CUM_TRST_DLY: Cumulative transit delay
   This item specifies the cumulative transit delay in milliseconds in a call accept since the original call request.
   Data format: single–word value, in the range 0–65,535.

PSI$C_NCB_CUM_TRST_DLY_R: Cumulative transit delay
   This item specifies the cumulative transit delay in milliseconds for an incoming call since the original call request.
   Data format: single–word value, in the range 0–65,535.

PSI$C_NCB_DIAGCODE: Diagnostic code
   This item specifies a diagnostic code, originated either by your PSDN, or by a user application.
   Data format: single–byte value in the range 0–255.

PSI$C_NCB_DTECLASS: DTE Class
   This item specifies the DTE Class from which a DTE is selected to make the call. This item also points to the Gateway system which will take the call.
   Data format: counted string.

PSI$C_NCB_ETE_TRST_DLY: End–to–end transit delay
   This item specifies the acceptable transit delay in milliseconds for an outgoing call from one DTE to the next. This item is used with PSI$C_NCB_CUM_TRST_DLY.
   Data format: single–word value, in the range 0–65,535.

PSI$C_NCB_EXPEDITE: Negotiate use of expedited data (interrupts)
   This item specifies how X.25 for OpenVMS is to handle interrupts over a virtual circuit.
   0 = Interrupts not permitted.
   1 = Interrupts permitted.
   Data format: single–byte value, 0 or 1.
**PSI$C_NCB_FILTER: Filter entity**

Use this item to specify another filter when your program redirects an incoming call.

*Data format:* counted string.

**PSI$C_NCB_FLT_PRI: Destination priority**

This item specifies the priority of the destination that has received an incoming call.

*Data format:* single–word value in the range 0–65535.

**PSI$C_NCB_FLT_REDPRI: Redirection priority**

This item restricts the destinations that are to be searched when an incoming call is redirected. Only destinations with a priority lower than the specified priority are searched.

To ensure that your process is not searched again, move the priority value from PSI$C_NCB_FLT_PRI to PSI$C_NCB_FLT_REDPRI.

*Data format:* single–word value in the range 0–65535.

**PSI$C_NCB_FSEL: Fast select without restriction**

This item specifies the Fast Select facility, which allows a DTE to include a user data field in the call request. The receiving DTE can accept or reject the call request, and send user data with its response.

The PSDN must subscribe to this facility.

*Data format:* no data field.

**PSI$C_NCB_FSEL_RES: Fast select with restricted response**

This item specifies the Fast Select facility, which allows a DTE to include a user data field in the call request. The restriction prevents the receiving DTE from accepting the request to set up a virtual circuit, but allows user data to be sent with the rejection.

The PSDN must subscribe to this facility.

*Data format:* no data field.

**PSI$C_NCB_GATEWAY: Gateway identifier**

This item is reserved for future use.

If you use this item, X.25 for OpenVMS returns an error in the IOSB: the first word contains SS$_IVDEVNAM; the third word contains PSI$C_ERR_INVITEM.

**PSI$C_NCB_ICI: Incoming call identifier**

This item identifies the incoming call. Do not modify or specify this item.

*Data format:* longword.

**PSI$C_NCB_LOCDTE: Local DTE address**

This item specifies the address of the local DTE in incoming calls.

*Data format:* counted string. ♦
**PSI$C_NCB_LOCFAC: Local PSDN facilities (outgoing)**

Your program can specify coding for local PSDN facilities in this item when making an outgoing call request.

X.25 for OpenVMS copies the contents of this item into the facilities field of the call request packet, where they may appear in a different order from that specified in this item.

*Data format:* counted string. If required by the PSDN, you should include the local facilities marker.

**PSI$C_NCB_LOCFACR: Local PSDN facilities (incoming)**

Your program can specify coding for local PSDN facilities in this item when accepting or rejecting an incoming call request.

X.25 for OpenVMS copies the contents of this item into the facilities field of the call request packet, where they may appear in a different order to that specified in this item.

*Data format:* counted string. If required by the PSDN, you should include the local facilities marker.

**PSI$C_NCB_LOCSUBADR: Local subaddress**

X.25 for OpenVMS supplies this item for incoming calls.

*Data format:* counted string.

**PSI$C_NCB_MAX_TRST_DLY: Maximum acceptable transit delay**

This item specifies the maximum acceptable transit delay in milliseconds.

*Data format:* single-word value, in the range 0–65535.

**PSI$C_NCB_MIN_THRUCLS: Minimum throughput class**

This item specifies the minimum data rate for a virtual circuit.

*Data format:* single-byte value, in the range 0–15.

**PSI$C_NCB_NET_USER_ID: Network user identifier**

This item identifies a network user.

*Data format:* counted string, PSDN–specific format.

**PSI$C_NCB_NULL: Null item identifier**

Any item containing this code is ignored.

*Data format:* any format accepted.

**PSI$C_NCB_PKTSIZE: Packet size**

This item specifies the packet size for an outgoing call, where the packet size requested is different from the default for the PSDN. The packet size requested must be valid for the PSDN.

*Data format:* single-word value, in the range 16–4096. The value must be a power of 2.

**PSI$C_NCB_PVCNAM: PVC identifier**

This item specifies a Permanent Virtual Circuit.

*Data format:* counted string.
**PSIC_NCB_RCV_QUOTA:** Total size of receive buffers

This item specifies the total size of the buffers (in bytes) that X.25 for OpenVMS uses to hold received data that has not yet been read by your application.

Minimum value = \((\text{packet-size} + 276)\)

Maximum value = \((\text{packet-size} + 276) \times \text{window-size}\)

These buffer sizes are deducted from the BYTLM quota for your process.

*Data format:* longword value.

**PSIC_NCB_REASON:** Code for X.25 for OpenVMS clearing a call

X.25 for OpenVMS uses this item to specify a code indicating why it cleared a call. Table C–2 lists the possible symbolic values.

*Data format:* single-byte value in the range 0–255.

<table>
<thead>
<tr>
<th>Type Code</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSI$C_L3_NETWRK</td>
<td>PSDN initiated</td>
</tr>
<tr>
<td>PSI$C_L3_NETERR</td>
<td>PSDN protocol error</td>
</tr>
<tr>
<td>PSI$C_L3_LNKDN</td>
<td>Communications link failed</td>
</tr>
<tr>
<td>PSI$C_L3_LNKUP</td>
<td>Communications link operational</td>
</tr>
<tr>
<td>PSI$C_L3_LNKRT</td>
<td>Communications link restarted</td>
</tr>
<tr>
<td>PSI$C_L3_GATDISC</td>
<td>Connection to Gateway (Multihost node) disconnected</td>
</tr>
<tr>
<td>PSI$C_L3_NETDISC</td>
<td>Connection to Gateway (Multihost node) lost</td>
</tr>
<tr>
<td>PSI$C_L3_LOCMGT</td>
<td>Network management function</td>
</tr>
<tr>
<td>PSI$C_L3_CALCOL</td>
<td>Call collision</td>
</tr>
<tr>
<td>PSI$C_L3_NETTIM</td>
<td>Timeout on network</td>
</tr>
</tbody>
</table>

**PSIC_NCB_REMDTE:** Remote DTE address

This item specifies the address of the remote DTE to which the call is to be made.

*Data format:* counted string.

**PSIC_NCB_REMSUBADR:** Remote subaddress

This item specifies the subaddress of the remote DTE to which the call is to be made.

*Data format:* counted string. The length of this item depends on the PSDN.

**PSIS_NCB_RESPDATA:** Fast select response data

Your program can add this item when it accepts or rejects an incoming call that specifies the Fast Select facility.

The PSDN must subscribe to this facility.

*Data format:* counted string.

**PSIC_NCB_REVCHG:** Reverse charging request

This item specifies reverse charging on an outgoing call request.

*Data format:* no data field.
**PSI$C_NCB_RPOA: Remote Port Of Access**

Your program can specify a PSDN–specific code in this item to specify how a call is to be routed across international networks.

*Data format:* counted string of a multiple of four ASCII characters, each representing a value from 0 to 9.

**PSI$C_NCB_TEMPLATE: Template**

Specifies the template used for making the outgoing call.

*Data format:* counted string.

**PSI$C_NCB_THRUCLS: Throughput class**

This item specifies the maximum data rate for a virtual circuit.

*Data format:* single-byte value, in the range 0–255.

**PSI$C_NCB_TRANSIT_DELAY: Requested maximum transit delay**

This item specifies the transit delay, in milliseconds, of an outgoing call.

*Data format:* single-word value, in the range 0–65,535.

**PSI$C_NCB_USERDATA: User data field**

For an outgoing call request, this item specifies data to be passed to the remote DTE. However, some character positions may have significance for your PSDN. Refer to the technical guide for your PSDN for details.

For an incoming call request, this item specifies data originated by the remote DTE. This item is ignored if it appears in the NCB for accepting or rejecting an incoming call.

For a fast select call (incoming or outgoing), this item specifies user data to be included in the clear request packet when clearing the call.

*Data format:* counted string.

**PSI$C_NCB_WINSIZE: Window size**

This item specifies the window size for an outgoing call, where the packet size requested is different from the default for the PSDN. The window size requested must be valid for the PSDN.

*Data format:* single-word value, in the range 1–127.
C.6 Example NCB

The following example illustrates an NCB that you could use when issuing a fast select request to set up a virtual circuit. The example identifies a remote DTE (234219876543), a network (PSS), and includes a remote DTE subaddress (26). As the NCB specifies fast select, there is also a user data field (containing the string DATADATA).

```
OPEN_INFO_START:
  REMOTE_DTE:
    .WORD REMOTE_DTE_LENGTH
    .WORD PSISC_NCB_REMDTE
    .ASCIC /234219876543/
  REMOTE_DTE_LENGTH = .-REMOTE_DTE
  DTECLASS:
    .WORD DTECLASS_LENGTH
    .WORD PSISC_NCB_DTECLASS
    .ASCIC /PSS/
  DTECLASS_LENGTH = .-DTECLASS
  REMOTE_DTE_SUB:
    .WORD REMOTE_DTE_SUBLEN
    .WORD PSISC_NCB_REMSUBADR
    .ASCIC /26/
  REMOTE_DTE_SUBLEN = .-REMOTE_DTE_SUB
  OPEN_DATA:
    .WORD OPEN_DATA_LENGTH
    .WORD PSISC_NCB_USERDATA
    .ASCIC /DATADATA/
  OPEN_DATA_LENGTH = .-OPEN_DATA
  FAST_SELECT:
    .WORD FAST_SELECT_LENGTH
    .WORD PSISC_NCB_FSEL
  FAST_SELECT_LENGTH = .-FAST_SELECT
  OPEN_INFO_LENGTH = .-OPEN_INFO_START
```
If your program is to handle incoming calls, you must associate a mailbox with the channel to the NW unit. X.25 uses the mailbox to inform your program when incoming calls arrive, and when interrupts and other network events occur; for example, call has been cleared, network failure, remote DTE failure.

For incoming calls, your program should read the NCB in the mailbox associated with the channel that received the call. Your program must quote the incoming call identifier (PSIC_NCB_ICI) to accept or reject an incoming call.

If your program makes only outgoing calls, you need not use a mailbox. However, without using the mailbox, you cannot receive notification of interrupts and other network events.

**D.1 Format**

Figure D–1 shows the general format of a mailbox message.
The contents of a mailbox are as follows:

**MSGTYPE** (2 bytes) This code indicates the type of INFO. Table D–1 and Table D–2 detail the types of message that may be found in a mailbox.

**UNIT** (2 bytes) This is the binary number of the device unit to which the message applies. For MSG$_{CONNECT}$, the device unit number is zero.

**COUNT1** (1 byte) This is a count of the characters in NAME.

**NAME** (15 bytes) This is a counted string of up to 15 characters, giving the name of the device to which the message applies: NVA for X.29 calls, NWA for X.25 calls.

**COUNT2** (1 byte) This is a count of the characters in INFO. When the INFO field contains an NCB, the length of the NCB should be determined by subtracting the length of the MSGTYPE, UNIT, COUNT1, NAME, and COUNT2 fields from the total mailbox message length, which is placed in the second word of the IO status block. (Since an NCB can be more than 255 bytes in length, the COUNT2 field may not reflect the NCB length accurately.)

**INFO** This is a counted string. The contents depend on the message type, and are interpreted as shown in Table D–1 and Table D–2.

The value of byte 3 (reason for reset byte) of INFO for MSG$_{RESET}$ is one of the following:

- PSI$C_L3_NETWRKInitiated by the PSDN
- PSI$C_L3_NETERRPSDN protocol error
- PSI$C_L3_LNKDWNLink down
- PSI$C_L3_LNKUPLink up
- PSI$C_L3_LNKRTLink restarted
- PSI$C_L3_LOCMGTNetwork management function

Mailbox messages for X.25 and X.29 programming are shown in Table D–1 and Table D–2.
### Table D–1  Mailbox Message Types for X.25 Programming

<table>
<thead>
<tr>
<th>Type Code</th>
<th>Meaning</th>
<th>Mailbox Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG$_INTMSG</td>
<td>Interrupt message</td>
<td>Interrupt byte</td>
</tr>
<tr>
<td>MSG$_CONNECT</td>
<td>Incoming call/Call confirm</td>
<td>NCB</td>
</tr>
</tbody>
</table>
| MSG$_RESET      | Request to reset the virtual circuit | Byte 1: Diagnostic code  
|                 |                                  | Byte 2: Cause code  
|                 |                                  | Byte 3: Reason for reset  |
| MSG$_DISCON     | Either of:                        | NCB                 |
|                 | • Incoming request to clear the virtual circuit |               |
|                 | • Completion of outgoing request to clear the virtual circuit |               |
| MSG$_INCDAT     | Unsolicited incoming data available | (Not used) |
| MSG$_PATHLOST   | Line restart (PVC only)           | (Not used)          |
| MSG$_NETSHUT    | DECnet has shut down (Network processes only) | (Not used)          |

### Table D–2  Mailbox Message Types for X.29 Programming

<table>
<thead>
<tr>
<th>Type Code</th>
<th>Meaning</th>
<th>Mailbox Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG$_CONNECT</td>
<td>Incoming call confirm</td>
<td>NCB</td>
</tr>
<tr>
<td>MSG$_DISCON</td>
<td>Incoming call reject</td>
<td>NCB</td>
</tr>
<tr>
<td>MSG$_TRMUNSOLIC</td>
<td>Incoming call</td>
<td>NCB</td>
</tr>
</tbody>
</table>

Mailbox Messages  D–3
D.2 Mailbox Message Sizes

Caution

You should ensure that the maximum message size of each mailbox is large enough to hold the expected mailbox messages.

For an Access application of type X25 and X29, X.25 for OpenVMS creates a mailbox in which to place the incoming call details (the application locates this mailbox via the logical SYS$NET). By default, X.25 for OpenVMS creates this mailbox with a maximum message size of 512 bytes, which should be adequate for most configurations. However, if the incoming call NCB contains Filter and DTE Class names having a total size of more than 200 bytes, the default value for the mailbox’s maximum message size may need to be increased.

To increase this value, define the logical name X25$APPL_MBXMXMSG to be the required size. The logical must be defined in the “SYSTEM” logical name table, and must be defined before X.25 for OpenVMS is started. For example, add the following line to SYS$STARTUP:SYSTARTUP_VMS.COM before the command @SYS$STARTUP:X25$STARTUP.COM:

$ define/system/exec X25$APPL_MBXMXMSG 800

Note that the mailbox’s buffer quota may also need to be increased. The default buffer quota given to mailboxes created by X.25 for OpenVMS is the value of the SYSGEN parameter DEFMBXBUFQUO—this may be increased by defining the logical name X25$APPL_MBXBUFQUO. For example, add the following line to SYS$STARTUP:SYSTARTUP_VMS before the command @SYS$STARTUP:X25$STARTUP.COM:

$ define/system/exec X25$APPL_MBXBUFQUO 1600

For more information on the relationship between mailbox message size and buffer quota, refer to the description of $CREMBX in the OpenVMS System Services Reference Manual.
This appendix describes the standard PAD parameters. Table E–1 lists the parameter codes and parameter numbers.

Table E–1  PAD Parameter Codes

<table>
<thead>
<tr>
<th>Parameter Number</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PSI$K_X29_PAR_ESCAPE</td>
</tr>
<tr>
<td>2</td>
<td>PSI$K_X29_PAR_ECHO</td>
</tr>
<tr>
<td>3</td>
<td>PSI$K_X29_PAR_FORWARD</td>
</tr>
<tr>
<td>4</td>
<td>PSI$K_X29_PAR_TIMEOUT</td>
</tr>
<tr>
<td>5</td>
<td>PSI$K_X29_PAR_HOSTSYNC</td>
</tr>
<tr>
<td>6</td>
<td>PSI$K_X29_PAR_MESSAGES</td>
</tr>
<tr>
<td>7</td>
<td>PSI$K_X29_PAR_BREAK</td>
</tr>
<tr>
<td>8</td>
<td>PSI$K_X29_PAR_DISCARD</td>
</tr>
<tr>
<td>9</td>
<td>PSI$K_X29_PAR_CRFILL</td>
</tr>
<tr>
<td>10</td>
<td>PSI$K_X29_PAR_WRAP</td>
</tr>
<tr>
<td>11</td>
<td>PSI$K_X29_PAR_SPEED</td>
</tr>
<tr>
<td>12</td>
<td>PSI$K_X29_PAR_TTSYNC</td>
</tr>
<tr>
<td>13</td>
<td>PSI$K_X29_PAR_NEW_LINE</td>
</tr>
<tr>
<td>14</td>
<td>PSI$K_X29_PAR_LFFILL</td>
</tr>
<tr>
<td>15</td>
<td>PSI$K_X29_PAR_EDIT</td>
</tr>
<tr>
<td>16</td>
<td>PSI$K_X29_PAR_DELETE</td>
</tr>
<tr>
<td>17</td>
<td>PSI$K_X29_PAR_LINE_DELETE</td>
</tr>
<tr>
<td>18</td>
<td>PSI$K_X29_PAR_REDISPLAY</td>
</tr>
<tr>
<td>19</td>
<td>PSI$K_X29_PAR_DISPLAY_EDIT</td>
</tr>
<tr>
<td>20</td>
<td>PSI$K_X29_PAR_RESTRICT_ECHO</td>
</tr>
<tr>
<td>21</td>
<td>PSI$K_X29_PAR_PARITY</td>
</tr>
<tr>
<td>22</td>
<td>PSI$K_X29_PAR_PAGE_WAIT</td>
</tr>
</tbody>
</table>
Parameter 1 (PSISK_X29_PAR_ESCAPE)
This specifies the ASCII character that the PAD function uses as the Escape character. For example, a parameter value of 33 specifies ! as the Escape character. Specify one of the following values:

0  PAD does not enter command mode on receiving an Escape character.
1  PAD enters command mode on receiving the Escape character [Ctrl/P].
2 to 31 Specify the Escape character. These values are extensions to the CCITT values.
32 to 127 Specify the Escape character.

Parameter 2 (PSISK_X29_PAR_ECHO)
This specifies whether the PAD echoes the input entered at the X.29 terminal. Refer to the HP X.25 for OpenVMS—Utilities Guide for details of the relationship between PAD echoing and the OpenVMS local–echo characteristic (SET TERMINAL/LOCAL_ECHO). Specify one of the following values:

0  No PAD echo.
1  PAD echo.

Parameter 3 (PSISK_X29_PAR_FORWARD)
This specifies the characters that cause data to be transmitted from the PAD to the remote DTE. The CCITT values are 0, 2, 6 (2 + 4), 18 (2 + 16) and 126 (2 + 4 + 8 + 16 + 32 + 64).

0  Forward data whenever a packet is full.
1  Any alphanumeric character (from A to Z, a to z, or 0 to 9).
2  CR
4  ESC, BEL, ENQ, ACK
8  DEL, CAN, DC2
16  EXT, EOT
32  HT, LF, VT, FF
64  Any control character not given previously.

Parameter 4 (PSISK_X29_PAR_TIMEOUT)
This specifies the Timeout value for forwarding data.

0  Timeout.
1 to 255  Timeout value in units of 1/20 second.

Note that if parameter 15 is set to 1, timeouts are disabled.
Parameter 5 (PSI$K_X29_PAR_HOSTSYNC)
This specifies whether the PAD sends XON and XOFF control characters to the X.29 terminal in data transfer mode. The host-based PAD treats parameter values 1 and 2 in the same way. Specify one of the following values:
0  No device control.
1  XON/XOFF device control in data transfer mode.
2  XON/XOFF device control in both command mode and data transfer mode.

Parameter 6 (PSI$K_X29_PAR_MESSAGES)
This specifies whether messages from the PAD are sent to the X.29 terminal. Permitted values are any combination of the following:
0  PAD messages suppressed.
1  PAD messages transmitted.
4  (In command mode) PAD prompt transmitted.
8  PAD messages in nonstandard format. (This value is ignored by the host-based PAD.)

Parameter 7 (PSI$K_X29_PAR_BREAK)
This specifies the actions taken by the PAD when the user presses [Break]. The OpenVMS Terminal Driver does not recognize [Break], and so the host-based command PAD BREAK is used to simulate the Break action. Legal values are any combination of the following:
0  No action.
1  PAD sends an Interrupt to the remote DTE.
2  PAD sends a Reset to the remote DTE.
4  PAD sends Indication–of–break to the remote DTE.
8  PAD enters command mode.
16 PAD discards output to the terminal. This setting automatically sets PAD Parameter 8 (PSI$K_X29_PAR_DISCARD) to value 1 (discard data from remote DTE).

Parameter 8 (PSI$K_X29_PAR_DISCARD)
This specifies whether the PAD sends output from NV to the X.29 terminal. This performs a similar function to [Ctrl/O]. However whereas [Ctrl/O] requests NV to discard all output to the PAD, this parameter, if set, requests the PAD to discard the output received from NV. Hence, for reasons of efficiency and cost, [Ctrl/O] is preferred.
Specify one of the following values:
0  Normal data delivery.
1  Discard data from remote DTE.
**Parameter 9 (PSISK_X29_PAR_CRFILL)**
This specifies the number of padding characters after a Return. Permitted values are between 0 and 255. Specify a value between 0 and 7.

**Parameter 10 (PSISK_X29_PAR_WRAP)**
This specifies the character position where the PAD inserts a Return. For example, setting a value of 80 requests the PAD to insert a Return after the 80th character, and continue the text that follows on a new line.

This parameter corresponds to the /WRAP and /WIDTH characteristics of the terminal. Set this parameter to 0 for proper operation of X.29 terminals on OpenVMS. Otherwise, specify one of the following values:

0 No wraparound.
1 to 255 Specifies the maximum line length that the PAD will print or echo.

**Parameter 11 (PSISK_X29_PAR_SPEED)**
This is a read-only parameter. The setting corresponds to the /SPEED characteristic of the terminal. Do not attempt to set this parameter.

The parameter takes the following values:

<table>
<thead>
<tr>
<th>Speed (bits/s)</th>
<th>Parameter Value</th>
<th>Speed (bits/s)</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>10</td>
<td>1200</td>
<td>3</td>
</tr>
<tr>
<td>75</td>
<td>5</td>
<td>75/1200</td>
<td>11</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>1800</td>
<td>7</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>2400</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4800</td>
<td>13</td>
</tr>
<tr>
<td>134.5</td>
<td>1</td>
<td>9600</td>
<td>14</td>
</tr>
<tr>
<td>150</td>
<td>6</td>
<td>19200</td>
<td>15</td>
</tr>
<tr>
<td>200</td>
<td>8</td>
<td>48000</td>
<td>16</td>
</tr>
<tr>
<td>300</td>
<td>2</td>
<td>56000</td>
<td>17</td>
</tr>
<tr>
<td>600</td>
<td>4</td>
<td>64000</td>
<td>18</td>
</tr>
</tbody>
</table>

**Parameter 12 (PSISK_X29_PAR_TTSYNC)**
This specifies whether the PAD responds to XON and XOFF control characters sent from the X.29 terminal. Specify one of the following values:

0 No flow control.
1 Flow control on: the PAD responds to XON and XOFF control characters from the terminal.
Parameter 13 (PSISK_X29_PAR_NEW_LINE)
This specifies whether the PAD sends a LF with every Return received from the X.29 terminal for transmission to NV or received from NV for forwarding to the X.29 terminal.

Set this parameter to 0 for correct operation of X.29 terminals on OpenVMS. Otherwise specify one, or a combination, of the following values:

0  No LF inserted after CR.
1  LF inserted after every CR that is transmitted as data to the terminal.
2  LF inserted after every CR that is received as data from the terminal.
4  LF inserted after every CR that is echoed as data to the terminal.

Parameter 14 (PSISK_X29_PAR_LFFILL)
This specifies the number of padding characters the PAD sends after a LF. Permitted values are as for PSI$K_X29_PAR_CRFILL.

Parameter 15 (PSISK_X29_PAR_EDIT)
This parameter controls whether the PAD performs local editing when in data transfer mode.

You should set this parameter only when the terminal is set to /LOCAL_ECHO. Specify one of the following values:

0  No local editing in data transfer mode.
1  Editing dependent on PAD parameters 16, 17, 18. This setting disables the idle timer.

Parameter 16 (PSISK_X29_PAR_DELETE)
This specifies the ASCII character that the PAD uses as the Delete character. This parameter is ignored unless parameter 15 is set. Specify a value between 0 and 127:

0  No delete character allowed.
1 to 127  ASCII character code to be used for delete.

Parameter 17 (PSISK_X29_PAR_LINE_DELETE)
This specifies the ASCII character that the PAD uses as the Line Delete character. This parameter is ignored unless parameter 15 is set. Specify a value between 0 and 127.

Parameter 18 (PSISK_X29_PAR_REDISPLAY)
This specifies the ASCII character that the PAD uses as the Line Redisplay character. This parameter is ignored unless parameter 15 is set. Specify a value between 0 and 127.
Parameter 19 (PSISK_X29_PAR_DISPLAY_EDIT)
This controls the type of line editing display, as follows:
0  No display for PAD editing.
1  Hardcopy type.
2  Video terminal type.
3 to 127  ASCII character used to display editing.

Parameter 20 (PSISK_X29_PAR_RESTRICT_ECHO)
This specifies which characters are not echoed. Legal values are any combination of the following:
0  All characters echoed.
1  \r
2  \n
4  \vt, \ht, \ff
8  \bel, \bs
16  \esc, \enq
32  \ack, \nak, \stx, \soh, \eot, \etb, \etx
64  The editing characters (as set by PSISK_X29_PAR_DELETE, PSISK_X29_PAR_LINE_DELETE, PSISK_X29_PAR_REDISPLAY_LINE).
128  \del, and all other control characters.

Parameter 21 (PSISK_X29_PAR_PARITY)
This controls parity generation and checking. Legal values are any combination of the following:
0  None.
1  Parity checking.
2  Parity generation.

Parameter 22 (PSISK_X29_PAR_PAGE_WAIT)
This controls whether the PAD holds the display at the end of each page. Specify one of the following values:
0  Page wait disabled.
1 to 255  Number of lines to display before waiting.
A number of X.25 and X.29 example programs are provided in the SYS$EXAMPLES directory.

Table F–1 describes the example programs supplied on OpenVMS I64 and OpenVMS Alpha systems. Table F–2 summarizes the languages for which example programs are provided.

**Table F–1 Programming Examples (OpenVMS I64 and OpenVMS Alpha)**

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X25$CHARGING</td>
<td>An example of a charging program to analyze X.25 accounting records</td>
</tr>
<tr>
<td>X25$RECEIVE</td>
<td>One of a pair of demonstration programs that transfer data entered at the terminal from one OpenVMS system to another over a Packet Switching Data Network. See also X25$SEND.</td>
</tr>
<tr>
<td>X25$SEND</td>
<td>One of a pair of demonstration programs that transfer data entered at the terminal from one OpenVMS system to another over a Packet Switching Data Network. See also X25$RECEIVE.</td>
</tr>
<tr>
<td>X25$X29_DESTINATION</td>
<td>A simple example of a program in which an X.29 destination prompts the X.29 user for a password before allowing them to log in.</td>
</tr>
</tbody>
</table>

**Table F–2 Program/Language Matrix (OpenVMS I64 and OpenVMS Alpha)**

<table>
<thead>
<tr>
<th>Example Program</th>
<th>C</th>
<th>COBOL</th>
<th>FORTRAN</th>
<th>MACRO</th>
<th>PASCAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>X25$CHARGING</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>X25$RECEIVE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X25$SEND</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X25$X29_DESTINATION</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table F–3 describes the example programs supplied on OpenVMS VAX systems. Table F–4 summarizes the languages for which example programs are provided.

### Table F–3  Programming Examples (OpenVMS VAX)

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSI$CHARGING</td>
<td>An example of a charging program to analyze X.25 accounting records</td>
</tr>
<tr>
<td>PSI$X25_RECEIVE</td>
<td>One of a pair of demonstration programs that transfer data entered at the terminal from one OpenVMS system to another over a Packet Switching Data Network. cf. PSI$X25_SEND.</td>
</tr>
<tr>
<td>PSI$X25_SEND</td>
<td>One of a pair of demonstration programs that transfer data entered at the terminal from one OpenVMS system to another over a Packet Switching Data Network. cf. PSI$X25_RECEIVE.</td>
</tr>
<tr>
<td>PSI$X29_DESTINATION</td>
<td>A simple example of a program in which an X.29 destination prompts the X.29 user for a password before allowing them to log in.</td>
</tr>
<tr>
<td>X25$X29_NETPROCESS</td>
<td>A simple example of a program in which an X.29 network process validates a password before allowing the X.29 caller to log in.</td>
</tr>
<tr>
<td>X25$X29_NV_UNIT_NUMBER</td>
<td>An example that determines the unit number of a device, given the device name.</td>
</tr>
<tr>
<td>X25$X29_OUTGOING</td>
<td>A simple example of an X.29 program that is run to establish an X.29 circuit from a host to a remote PAD.</td>
</tr>
</tbody>
</table>

### Table F–4  Program/Language Matrix (OpenVMS VAX)

<table>
<thead>
<tr>
<th>Example Program</th>
<th>BASIC</th>
<th>C</th>
<th>COBOL</th>
<th>FORTRAN</th>
<th>MACRO</th>
<th>PASCAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSI$CHARGING</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PSI$X25_RECEIVE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PSI$X25_SEND</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PSI$X29_DESTINATION</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSI$X29_NETPROCESS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>PSI$X29_NV_UNIT_NUMBER</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>PSI$X29_OUTGOING</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

♦
Access level item code, 2–33
$ASSIGN, 2–2 to 2–3
use for Permanent Virtual Circuits, 2–2
astadr, 1–4
astprm, 1–4

Bilateral Closed User Group NCB item, C–6
BREAK parameter, E–3
Buffer size
  for NV Action Descriptor Block, 4–4, 4–8
  for PAD parameters, 4–9
  for P$SI$X$K_X29_BREAK_ACTION, 4–4
  for P$SI$X$K_X29_HOLD_TIMER, 4–6
  for P$SI$X$K_X29_HOST_ECHO_PARAMS, 4–7
  for P$SI$X$K_X29_INT_ACTION, 4–8
  for P$SI$X$K_X29_LOCAL_ECHO_PARAMS, 4–9
  for P$SI$X$K_X29_TEMP_NOHANG, 4–12

Called address extension, 2–13, 2–25, 2–35
Called address extension NCB item, C–5
Called DTE address item code, 2–33
Called extension item code, 2–33
Calling address extension NCB item, C–5
Call redirection reason NCB item, C–5
Calls
  using $CANCEL to clear, 2–4
  using $QIO(IO$_ACCESS!IO$_M_ACCEPT) to accept, 2–23 to 2–27
  using $QIO(IO$_ACCESS!IO$_M_ABORT) to reject, 2–19 to 2–22
  using $QIO(IO$_ACCESS!IO$_M_REDIRECT) to reject, 2–28 to 2–30
  using $QIO(IO$_DEACCESS) to clear, 2–39 to 2–42
$CANCEL, 2–4
chan, 1–4
Channel
  using $ASSIGN to associate with a mailbox, 2–2
  using $DASSGN to deassign, 2–7
Channel number
  using $ASSIGN to obtain, 2–2
Charging information request NCB item, C–6
Charging units NCB item, C–5
Circuit
  using $QIO(IO$_ACCESS) to set up, 2–11 to 2–18
Clearing a virtual call with $CANCEL, 2–4
Closed User Group
  notes for using $QIO(IO$_ACCESS), 2–11
Closed User Group item code, 2–34
Closed User Group NCB item, C–6
Common system services, 2–1 to 2–46
Completion status information, 5–2
Condition handler, 5–2
Conventions
  syntax, 1–4
$CREMBX, 2–5 to 2–6
$CREPRC, 5–5
CRFILL parameter, E–3
CUG
  see Closed User Group
Cumulative transit NCB item, C–6

D
$DASSGN, 2–7
  virtual circuit not cleared, 4–12
Data
  using $QIO(IO$_READVBLK) to receive data, 2–43 to 2–44
  using $QIO(IO$_WRITEVBLK) to transmit data, 2–45 to 2–46
X.25 criteria for sending, 2–45
Data mask item code, 2–33
Data transfer mode
  local editing, E–5
DCE
  operation as, 2–39
Delete character, E–5
DELETE parameter, E–5
Destination priority NCB item, C–7
Diagnostic code NCB item, C–6
DISCARD parameter, E–3
DISPLAY_EDIT parameter, E–6
DTE Class item code, 2–34
DTE Class NCB item, C–6
Dynamic filters, 2–31

**E**

ECHO parameter, E–2

Editing
  in data transfer mode, E–5
EDIT parameter, E–5
efn, 1–4

Elapsed time NCB item, C–6
End–to–end transit delay NCB item, C–6

Error codes, 1–5 to 1–6
Error returns, 5–1
Escape character, E–2
ESCAPE parameter, E–2
Example NCB, C–11
Example programs, F–1 to F–2

Exception conditions, 5–2
  overriding default handling, 5–2
Expedited data NCB item, C–6

**F**

Failure conditions, 5–2
Fast select calls
  and $QIO(IO$_ACCESS), 2–13
Fast select NCB item, C–7
Fast select response NCB item, C–9

Filters
  using $QIO(IO$_ACPCONTROL) to add, 2–32
  using $QIO(IO$_ACPCONTROL) to delete, 2–32

Filter types
  dynamic, 2–31
  static, 2–31

Format
  of numeric status codes, 5–3
  of system services, 1–4

Forward character, E–2
Forwarding
  timeout, E–2
FORWARD parameter, E–2

**G**

Gateway identifier NCB item, C–7
GETDVI, 2–8 to 2–10
  to get NW or NV unit number, 2–9
  to get the remote DTE address, 2–10
  using to connect NV device to VT device, 2–9

**H**

Hangup, 2–7
Hangup PAD parameter template
  using $QIO(IO$_NETCONTROL, PSI$K_X29_ 
  SET, PSI$K_X29_HANGUP_PARAMS) to set, 4–20

Hangup template
  using $QIO(IO$_NETCONTROL, PSI$K_X29_ 
  READ, PSI$K_X29_HANGUP_PARAMS) to read, 4–5

Hold Timer
  using $QIO(IO$_NETCONTROL, PSI$K_X29_ 
  READ, PSI$K_X29_HOLD_TIMER) to read, 4–6
  using $QIO(IO$_NETCONTROL, PSI$K_X29_ 
  SET, PSI$K_X29_HOLD_TIMER) to set, 4–23

Host–echo PAD parameter template
  using $QIO(IO$_NETCONTROL, PSI$K_ 
  X29_READ, PSI$K_X29_HOST_ECHO_ 
  PARAMS) to return, 4–7
  using $QIO(IO$_NETCONTROL, PSI$K_X29_ 
  SET, PSI$K_X29_HOST_ECHO_PARAMS) to set, 4–21 to 4–22

HOSTSYNC parameter, E–3

**I**

I/O channel
  using $CREMBX to assign to mailbox, 2–5
I/O Status Block, 1–5, 5–1 to 5–2
  address, 1–4
  byte count, 5–2
  completion status information, 5–2
  status codes, 1–3 to 1–6

Incoming call identifier NCB item, C–7

Incoming calls
  mailbox, D–1
  using $QIO(IO$_ACCESS!!IO$_M_ACCEPT) to accept, 2–23 to 2–27
  using $QIO(IO$_ACCESS!!IO$_M_ABORT) to reject, 2–19 to 2–22
  using $QIO(IO$_ACCESS!!IO$_M_REDIRECT) to redirect, 2–28 to 2–30

Incoming DTE item code, 2–33

Indication–of–break
  using $QIO(IO$_NETCONTROL, PSI$K_X29_ 
  SET, PSI$K_X29_BREAK_ACTION) to set, 4–18 to 4–19

Interrupt action
  using $QIO(IO$_NETCONTROL, PSI$K_X29_ 
  READ, PSI$K_X29_INT_ACTION) to return for X.29 terminal, 4–8
  using $QIO(IO$_NETCONTROL, PSI$K_X29_ 
  SET, PSI$K_X29_INT_ACTION) to set for X.29 terminal, 4–24
Interrupts
length, 3–4
using $QIO(IO$_NETCONTROL, PSI$K_ INTACK) to acknowledge for X.25 calls, 3–2 to 3–3
using $QIO(IO$_NETCONTROL, PSI$K_ INTERRUPT) to transmit for X.25 calls, 3–4 to 3–5
IO$_NETCONTROL, PSI$K_X29_READ_SPECIFIC, 4–13 to 4–15
iosb, 1–4
IOSB
see I/O Status Block
Item codes, 2–33 to 2–34, C–2 to C–4
Items
in network process declaration block, 2–32

L
LFFILL parameter, E–5
Line Delete character, E–5
Line editing display, E–6
Linefeed, E–4
Line Redisplay character, E–5
LINE_DELETE parameter, E–5
Local DTE address NCB item, C–7
Local–echo PAD parameter template
using $QIO(IO$_NETCONTROL, PSI$K_X29_READ, PSI$K_X29_LOCAL_ECHO_PARAMS) to read, 4–9
Local–echo parameters
using $QIO(IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_LOCAL_ECHO_PARAMS) to set for X.29 terminal, 4–25 to 4–26
Local PSDN facilities NCB item, C–8
Local subaddress NCB item, C–8

M
Mailbox, D–1 to D–4
assigning an I/O channel, 2–5
using $ASSIGN to associate with a channel, 2–2
using $CREMBX to create, 2–5
Mailbox message types
X.25 programming, D–3
X.29 programming, D–3
Maximum acceptable transit delay NCB item, C–8
MESSAGES item code, E–3
Messages to X.29 terminal, E–3
Minimum throughput class NCB item, C–8
Modifiers for IO$_WRITEVBLK, 2–45
Modifying called line address NCB item, C–5

N
Name of filter item code, 2–34
NCB
see Network Connect Block
NCB data type formats
counted string, C–1
longword, C–1
single byte, C–1
single word, C–1
Network Connect Block, C–1 to C–11
data type formats, C–1
element, C–11
incoming call request, C–2
item codes, C–2 to C–4
item descriptions, C–5 to C–10
item functions, C–2
outgoing call request, C–2
Permanent Virtual Circuits, C–2
Switched Virtual Circuits, C–2
use, C–1
Network process
using $QIO(IO$_ACP CONTROL) to declare, 2–31 to 2–38
Network process declaration block, 2–31 to 2–35
item codes, 2–33 to 2–34
items, 2–32
Network user identifier NCB item, C–8
NEW_LINE parameter, E–4
Nonstandard PAD parameters
using $QIO(IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_PAD_PARAMS) to set, 4–27
NUI
see Network user identifier
Null item identifier NCB item, C–8
NV Action Descriptor Block
using $QIO(IO$_NETCONTROL, PSI$K_X29_READ, PSI$K_X29_BREAK_ACTION) to return break action, 4–4
using $QIO(IO$_NETCONTROL, PSI$K_X29_READ, PSI$K_X29_INT_ACTION) to return interrupt action, 4–8
NV device
QIO functions to, 4–1
NV terminal characteristics
using $QIO(IO$_NETCONTROL, PSI$K_X29_READ) to read, 4–3 to 4–12
using $QIO(IO$_NETCONTROL, PSI$K_X29_SET) to set, 4–16 to 4–31
NV unit number
using $GETDVI to obtain, 2–9
NW unit number
using $GETDVI to obtain, 2–9
O
Object process NCB item, C–7
Outgoing calls

mailbox, D–1

P
Packet size NCB item, C–8
PAD
echo, E–2
Padding characters, E–3, E–5
PAD parameters, E–1 to E–6
codes, 4–27, E–1
list, 4–27
nonstandard, 4–27
standard, E–1
using $QIO(IO\_NETCONTROL, PSI\_K\_X29\_READ) to read, 4–3 to 4–12
using $QIO(IO\_NETCONTROL, PSI\_K\_X29\_READ, PSI\_K\_X29\_PAD\_PARAMS) to read, 4–10 to 4–11
using $QIO(IO\_NETCONTROL, PSI\_K\_X29\_READ\_SPECIFIC) to read specific, 4–13 to 4–15
using $QIO(IO\_NETCONTROL, PSI\_K\_X29\_READ\_SPECIFIC, PSI\_K\_X29\_PAD\_PARAMS) to read specific, 4–14 to 4–15
using $QIO(IO\_NETCONTROL, PSI\_K\_X29\_SET, PSI\_K\_X29\_PAD\_PARAMS) to set, 4–27 to 4–28
PAD reselection message
using $QIO(IO\_NETCONTROL, PSI\_K\_X29\_SET, PSI\_K\_X29\_PAD\_RESELECTION) to send, 4–29 to 4–30
PAGE\_WAIT parameter, E–6
Parameters
see PAD parameters
Parity
checking, E–6
generation, E–6
item code, E–6
Permanent Virtual Circuit identifier NCB item, C–8
Permanent Virtual Circuits
construction of NCB, C–2
use of $ASSIGN, 2–2
Priority
for $QIO(IO\_ACCESS!IO\_M\_REDIRECT), 2–28
Priority item code, 2–34
Privileges
to use $QIO(IO\_ACCESS!IO\_M\_ABORT), 2–19
to use $QIO(IO\_ACCESS!IO\_M\_ACCEPT), 2–23
Privileges (cont’d)
to use $QIO(IO\_ACCESS!IO\_M\_REDIRECT), 2–28
to use $QIO(IO\_ACCESS), 2–11
to use $QIO(IO\_ACP\_CONTROL), 2–35
Process execution modes
resource wait, 5–4
system service failure exception, 5–4
Process quotas, 5–4
Programming examples, F–1 to F–2
PSDN clearing code NCB item, C–5
PSI\$DECLNAME rights identifier, 2–35
PSI\$X25\_USER rights identifier, 2–11, 2–19, 2–23, 2–28

Q
QIO arguments, 1–4
QIOs
IO\_ACCESS, 2–11 to 2–30
IO\_ACCESS!IO\_M\_ABORT, 2–19 to 2–22
IO\_ACCESS!IO\_M\_ACCEPT, 2–23 to 2–27
IO\_ACCESS!IO\_M\_REDIRECT, 2–28 to 2–30
IO\_ACP\_CONTROL, 2–31 to 2–38
IO\_DEACCESS, 2–39 to 2–42
IO\_NETCONTROL, 3–2 to 4–31
IO\_NETCONTROL, PSI\_K\_INTACK, 3–2 to 3–3
IO\_NETCONTROL, PSI\_K\_INTERRUPT, 3–4 to 3–5
IO\_NETCONTROL, PSI\_K\_RESET, 3–6 to 3–7
IO\_NETCONTROL, PSI\_K\_RESTART, 3–8
IO\_NETCONTROL, PSI\_K\_X29\_READ, 4–3 to 4–12
IO\_NETCONTROL, PSI\_K\_X29\_READ, PSI\_K\_X29\_BREAK\_ACTION, 4–4
IO\_NETCONTROL, PSI\_K\_X29\_READ, PSI\_K\_X29\_HANGUP\_PARAMS, 4–5
IO\_NETCONTROL, PSI\_K\_X29\_READ, PSI\_K\_X29\_HOLD\_TIMER, 4–6
IO\_NETCONTROL, PSI\_K\_X29\_READ, PSI\_K\_X29\_HOST\_ECHO\_PARAMS, 4–7
IO\_NETCONTROL, PSI\_K\_X29\_READ, PSI\_K\_X29\_INT\_ACTION, 4–8
IO\_NETCONTROL, PSI\_K\_X29\_READ, PSI\_K\_X29\_LOCAL\_ECHO\_PARAMS, 4–9
IO\_NETCONTROL, PSI\_K\_X29\_READ, PSI\_K\_X29\_PAD\_PARAMS, 4–10 to 4–11
IO\_NETCONTROL, PSI\_K\_X29\_READ, PSI\_K\_X29\_TEMP\_NOHANG, 4–12
IO\_NETCONTROL, PSI\_K\_X29\_READ\_SPECIFIC, 4–13 to 4–15
IO\_NETCONTROL, PSI\_K\_X29\_READ\_SPECIFIC, PSI\_K\_X29\_PAD\_PARAMS, 4–14 to 4–15
IO\_NETCONTROL, PSI\_K\_X29\_SET, 4–16, 4–31

Index–4
QIOs (cont’d)

IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_BREAK_ACTION, 4–18 to 4–19

IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_HANGUP_PARAMS, 4–20

IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_HOLD_TIMER, 4–23

IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_HOST_ECHO_PARAMS, 4–21 to 4–22

IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_INT_ACTION, 4–24

IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_LOCAL_ECHO_PARAMS, 4–25 to 4–26

IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_PAD_PARAMS, 4–27 to 4–28

IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_PAD_RESELECTION, 4–29 to 4–30

IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_TEMP_NOHANG, 4–31

IO$_READVBLK, 2–43 to 2–44

IO$_WRITEVBLK, 2–45 to 2–46

NV device, 4–1

return status codes, 1–5 to 1–6

terminal driver, B–2

Quotas, 5–4

Resets

using $QIO(IO$_NETCONTROL, PSI$K_RESET) to confirm receipt for X.25 calls, 3–6 to 3–7

Resources

response of program to lack of, 5–4

Resource wait mode, 5–4

Restart

using $QIO(IO$_NETCONTROL, PSI$K_RESTART) to confirm receipt for X.25, 3–8

RESTRICT_ECHO parameter, E–6

Return status, 1–5 to 1–6, 5–1 to 5–5

testing, 5–3

Reverse charging NCB item, C–9

Rights identifier PSI$DECLNAME, 2–35

Rights identifier PSI$X25_USER, 2–11, 2–19, 2–23, 2–28

RPOA

see Remote Port Of Access

S

Secondary status

for $QIO(IO$_ACCESS), 2–16 to 2–18, 5–1 to 5–2

for $QIO(IO$_ACPCONTROL), 2–38, 5–1 to 5–2

for $QIO(IO$_DEACCESS), 2–41 to 2–42, 5–1 to 5–2

for $QIO(IO$_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_PAD_PARAMS), 5–1 to 5–2

for $QIO(IO$_READVBLK), 5–1 to 5–2

for $QIO(IO_NETCONTROL, PSI$K_X29_SET, PSI$K_X29_PAD_PARAMS), 2–44

Segment count NCB item, C–5

$SETRWM, 5–4

$SETSFM, 5–4

SET TERMINAL/X29 equivalent QIO, 4–16

SHOW TERMINAL/X29 equivalent QIO, 4–3

Specific PAD parameters

using $QIO(IO$_NETCONTROL, PSI$K_X29_READ_SPECIFIC) to read, 4–13 to 4–15

using $QIO(IO$_NETCONTROL, PSI$K_X29_READ_SPECIFIC, PSI$K_X29_PAD_PARAMS) to read, 4–14 to 4–15

SPEED parameter, E–4

Standard PAD parameters, E–1 to E–6

Static filters, 2–31

Status codes, 1–5 to 1–6, 5–1 to 5–5

Status values, 5–1 to 5–5

Subaddress high item code, 2–34

Subaddress low item code, 2–34

Switched Virtual Circuits

construction of NCB, C–2

Index–5
Symbolic codes, 5–3 to 5–5
Syntax
 for system services, 1–4
System dynamic memory, 5–4
System service calls
 transmission characteristics, B–2
System service failure exception mode, 5–4
 condition handling, 5–4
 error messages, 5–4
System services, 1–1 to 1–6, 5–1 to 5–5, A–1 to
 A–2, B–1 to B–2
 arguments, 1–4
 $ASSIGN, 2–2 to 2–3
 $CANCEL, 2–4
 common, 2–1
 completion, 1–5, 5–1 to 5–5
 $CREMBX, 2–5 to 2–6
 $DASSGN, 2–7
 $GETDVI, 2–8 to 2–10
 IO$_NETCONTROL, PSI$K_X29_READ_ SPECIFIC, 4–13 to 4–15
 $QIO, See also QIOs, 2–11 to 2–46
 summary of X.25, A–1 to A–2
 summary of X.29, B–1 to B–2
 syntax, 1–4
 X.25–specific, 3–1

T
Template NCB item, C–10
TEMP_NOHANG
 using $QIO(IO$_NETCONTROL, PSI$K_X29_ READ, PSI$K_X29_TEMP_NOHANG) to
 read flag, 4–12
 using $QIO(IO$NETWORK, PSI$K_X29_X29_ SET, PSI$K_X29_TEMP_NOHANG) to
 enable, 4–31
Terminal driver QIOs, B–2
Testing for failure and warning conditions, 5–2
Testing return status codes, 5–3
Throughput class NCB item, C–10
TIMEOUT parameter, E–2
Timeout value, E–2
Transmission characteristics, B–2
 system service calls, B–2
TTSYNC character, E–4
TTSYNC parameter, E–4

U
User data
 for $QIO(IO$ACCESS), 2–13
User data field NCB item, C–10
User data item code, 2–34
User group
 notes for using $QIO(IO$_ACCESS), 2–11

V
Virtual calls
 using $CANCEL to clear, 2–4
 using $QIO(IO$ACCESS!IO$M_ACCEPT) to
 accept, 2–23 to 2–27
 using $QIO(IO$ACCESS!IO$M_ABORT) to
 reject, 2–19 to 2–22
 using $QIO(IO$ACCESS!IO$M_REDIRECT) to
 reject, 2–28 to 2–30
Virtual circuit
 using $QIO(IO$_ACCESS) to set up, 2–11 to
 2–18
 using $QIO(IO$_DEACCESS) to clear, 2–39 to
 2–42
 using $QIO(IO$_NETCONTROL, PSI$K_ RESET) to reset for X.25 calls, 3–6 to 3–7

W
Wait state, 5–4
Warning conditions, 5–2
Warning returns, 5–1
Window size NCB item, C–10
WRAP parameter, E–4

X
X.25 clearing code NCB item, C–9
X.25 system service calls, A–1 to A–2
X.25 system services, 3–1 to 3–8
X.29 system service calls, B–1 to B–2
X.29 system services, 4–1 to 4–31
X25 Server filter
 item code, 2–33
X29 Server filter
 item code, 2–33
XOFF, E–3, E–4
XON, E–3, E–4