VMS Software

VSI X.25 for OpenVMS Programming Reference

Operating System and Version: VSI OpenVMS IA-64 Version 8.4-1H1 or higher
VSI OpenVMS Alpha Version 8.4-2L1 or higher

Software Version: VSI X.25 for OpenVMS Version 2.1

VMS Software, Inc. (VSI)
Boston, Massachusetts, USA

VSI X.25 for OpenVMS Programming Reference

Table of Contents

Preface

1. About VSI

VMS Software, Inc. (VSI) is an independent software company licensed by Hewlett Packard Enterprise to develop and support the OpenVMS operating system.

2. Intended 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.

3. Document 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:`.

4. Related Documents

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

VSI DECnet-Plus for OpenVMS Documentation

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

- *VSI OpenVMS DECnet-Plus Introduction and User's Guide* [https://docs.vmssoftware.com/vsi-openvms-decnet-plus-introduction-and-user-s-guide]

  This manual provides general information on DECnet-Plus and describes the concept of packet switching data networks.

- *VSI OpenVMS DECnet-Plus Installation and Configuration* [https://docs.vmssoftware.com/vsi-openvms-decnet-plus-installation-and-configuration]

  This manual describes how to install and configure VSI DECnet-Plus for OpenVMS software. For OpenVMS IA-64 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 IA-64 and OpenVMS Alpha systems are provided in the *VSI X.25 for OpenVMS Configuration Guide* [https://docs.vmssoftware.com/vsi-x-25-for-openvms-configuration-guide]. For OpenVMS VAX systems, this manual also describes how to install and configure the X.25 functionality provided by VSI DECnet-Plus for OpenVMS VAX.

- *VSI DECnet-Plus Network Management Guide* [https://docs.vmssoftware.com/vsi-decnet-plus-network-management-guide]

  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.

- *VSI OpenVMS DECnet-Plus Network Control Language Reference* [https://docs.vmssoftware.com/vsi-openvms-decnet-plus-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.

VSI X.25 for OpenVMS Documentation

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

- *VSI X.25 for OpenVMS Configuration Guide* [https://docs.vmssoftware.com/vsi-x-25-for-openvms-configuration-guide] (OpenVMS IA-64 and OpenVMS Alpha)

  This manual explains how to configure X.25 for OpenVMS software on OpenVMS IA-64 and OpenVMS Alpha systems.

- *VSI 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.

- *VSI X.25 for OpenVMS Problem Solving Guide* [https://docs.vmssoftware.com/vsi-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.

- *VSI X.25 for OpenVMS Programming Guide*

  This manual describes how to write X.25 and X.29 programs to perform network operations.

- *VSI 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 VSI X.25 for OpenVMS Programming.

- *VSI 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.

- *VSI X.25 for OpenVMS Installation Guide* [https://docs.vmssoftware.com/vsi-x-25-for-openvms-installation-guide/]

  This manual describes how to install VSI X.25 for OpenVMS V2.1 HPE servers running the OpenVMS operating system. This guide is intended for system managers who are responsible for installing VSI X.25 for OpenVMS V2.1.

- *VSI X.25 for OpenVMS Management Guide* [https://docs.vmssoftware.com/vsi-x-25-management-guide/]

  This manual provides information applicable to the X.25 functionality provided by VSI X.25 for OpenVMS and VSI DECnet-Plus for OpenVMS VAX.

- *VSI 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.

VSI OpenVMS Documentation

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

- *VSI OpenVMS User's Manual* [https://docs.vmssoftware.com/vsi-openvms-user-s-manual/].

- *VSI OpenVMS DCL Dictionary Part One* [https://docs.vmssoftware.com/vsi-openvms-dcl-dictionary-a-m] and *Part Two* [https://docs.vmssoftware.com/vsi-openvms-dcl-dictionary-n-z].

- *VSI OpenVMS System Management Utilities Reference Manual, Volume I* [https://docs.vmssoftware.com/vsi-openvms-system-management-utilities-reference-manual-volume-i-a-l] and *Volume II* [https://docs.vmssoftware.com/vsi-openvms-system-management-utilities-reference-manual-volume-ii-m-z].
- *HP OpenVMS System Services Reference Manual Part One* [https://docs.vmssoftware.com/vsi-openvms-system-services-reference-manual-a-getuai/] and *Part Two* [https://docs.vmssoftware.com/vsi-openvms-system-services-reference-manual-getutc-z/].
- *VSI OpenVMS Guide to System Security* [https://docs.vmssoftware.com/vsi-openvms-guide-to-system-security].

5. OpenVMS Documentation

The full VSI OpenVMS documentation set can be found on the VMS Software Documentation webpage at https://docs.vmssoftware.com.

6. VSI Encourages Your Comments

You may send comments or suggestions regarding this manual or any VSI document by sending electronic mail to the following Internet address: `<docinfo@vmssoftware.com>`. Users who have VSI OpenVMS support contracts through VSI can contact `<support@vmssoftware.com>` for help with this product.

7. 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 1. X.25 Terminology

VAX P.S.I.	X.25 for OpenVMS
VAX P.S.I.	X.25 for OpenVMS VAX
Access system	X.25 Client system
Native system	X.25 Direct Connect system
Multihost system	X.25 Connector system
Gateway system	X.25 Connector system

In addition to the terms shown in the previous table, 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 2. X.25 for OpenVMS Client/Server Terminology

Client system	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.)
Relay Client system	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.)

Relay–Client	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.
Relay system	An X.25 Direct Connect or Connector system with the X.25 Relay module enabled.
Server Client system	Another term for a Client system.
Server–Client	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.

For more information about clients, servers, and relays in X.25 for OpenVMS, refer to the *VSI X.25 for OpenVMS Configuration Guide* [https://docs.vmssoftware.com/vsi-x-25-for-openvms-configuration-guide] and the *VSI X.25 for OpenVMS Management Guide* [https://docs.vmssoftware.com/vsi-x-25-management-guide].

8. Conventions

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

Convention	**Meaning**
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 monospaced`** 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.

Convention	**Meaning**
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.

Chapter 1. Introduction

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

System Service	Description
Services Common to X.25 and X.29 Programming (Refer to Chapter 2)	
$ASSIGN	Assigns a Channel
$CANCEL	Clears a Virtual Call on a Channel
$CREMBX	Creates a Mailbox and Assigns a Channel
$DASSGN	Deassigns the Channel
$GETDVI	Gets the NV Device Number or Remote DTE Address
$QIO(IO$_ACCESS)	Sets Up a Virtual Circuit
$QIO(IO$_ACCESS!IO$M_ABORT)	Rejects a Call
$QIO(IO$_ACCESS!IO$M_ACCEPT)	Accepts a Call
$QIO(IO$_ACCESS!IO$M_REDIRECT)	Redirects a Call
$QIO(IO$_ACPCONTROL)	Declares a Network Process
$QIO(IO$_DEACCESS)	Clears a Virtual Circuit
$QIO(IO$_READVBLK)	Receives Data
$QIO(IO$_WRITEVBLK)	Transmits Data
Services Specific to X.25 Programming (Refer to Chapter 3)	
$QIO(IO$_NETCONTROL, PSI$K_INTACK)	Confirms Receipt of an Interrupt
$QIO(IO$_NETCONTROL, PSI $K_INTERRUPT)	Transmits an Interrupt

System Service	Description
$QIO(IO$_NETCONTROL, PSI$K_RESET)	Resets a Virtual Circuit or Confirms Receipt of a Reset
$QIO(IO$_NETCONTROL, PSI$K_RESTART)	Confirms Receipt of a Restart
Services Specific to X.29 Programming (Refer to Chapter 4)	
$QIO(IO$_NETCONTROL, PSI$K_X29_READ)	Reads X.29 Terminal Characteristics
$QIO(IO$_NETCONTROL, PSI $K_X29_READ_SPECIFIC)	Reads Specific X.29 Parameters
$QIO(IO$_NETCONTROL, PSI$K_X29_SET)	Sets X.29 Terminal Characteristics

1.1.1. Format of System Service Descriptions

In Chapter 2, Chapter 3, and Chapter 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 Chapter 2, Chapter 3, and Chapter 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.

- A character is one of the set of alphanumerics that includes:
    - A to Z
    - a to z
    - 0 to 9
    - _ (underscore)
    - $ (dollar)
- 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.
- Square brackets [ ] enclose optional keywords and arguments. Do not include the brackets when entering the system service.
- 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.

Chapter 2. Common System Services

Table 2.1 summarizes the system services common to both X.25 and X.29 programming.

Table 2.1. Common System Services

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

$ASSIGN

$ASSIGN — Assign a Channel

Purpose

$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 `NWAuu:` or `NVAuu:` (where *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 *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 **only one channel** for each virtual circuit to the `NWAuu` 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.

Format

$ASSIGN *devnam*,*chan*,[*acmode*],[*mbxnam*]

Arguments

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.
chan	Address of a word to receive the channel number assigned.
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.
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.

$CANCEL

$CANCEL — Clear a Virtual Call on a Channel

Purpose

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 canceled, 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 canceled 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`.

```
; 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.

$CREMBX

$CREMBX — Create a Mailbox and Assign a Channel

Purpose

$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 `MBAuu:`. 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.

$DASSGN

$DASSGN — Deassign the Channel

Purpose

$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.

$GETDVI

$GETDVI — Get NV Device Number or Remote DTE Address

Purpose

In X.25 programs, you can use $GETDVI to obtain the NW unit number (see the section called "$GETDVI - Get NW or NV Unit Number").

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

- Obtain the NW unit number (see the section called "$GETDVI - Get NW or NV Unit Number").
- Obtain the NV unit number (see the section called "$GETDVI - Get NW or NV Unit Number").
- Obtain the remote DTE address (see the section called "$GETDVI - Get Remote DTE Address of PAD Connected to NV").

$GETDVI - Get NW or NV Unit Number

Purpose

$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 *VSI 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`.

```
NV_CHAN:
  .BLKW  1                      ; Channel to NV
```

```
UNIT_LIST:
  .WORD  4                      ; length (in bytes) of buffer
                                ; for $GETDVI to output
  .WORD  DVI$_UNIT              ; item code
  .LONG  NV_UNIT                ; Address of buffer for $GETDVI
                                ; to output
  .LONG  NV_UNIT_LENGTH         ; Address of word for $GETDVI
                                ; to put the amount of data output
  .LONG 0                       ; End of the item-list
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.
    -                           ; wanted from system service.
    IOSB = IO_STATUS            ; Status return
  BSBW ERROR                    ; Check for error
```

Return Status

SS$_NORMAL	Service completed successfully.

$GETDVI - Get Remote DTE Address of PAD Connected to NV

Purpose

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
                                 ; for $GETDVI to output
  .WORD  DVI$_TT_ACCPORNAM       ; item code
  .LONG  PAD_REMDTE              ; Address of buffer for $GETDVI
                                 ; to output
  .LONG  PAD_REMDTE_LENGTH       ; Address of word for $GETDVI
                                 ; to put the amount of data output
  .LONG  0                       ; End of the item-list
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,-             ; channel
    ITMLST = REMDTE_LIST         ; Address of item-list of information.
    -                            ; wanted from system service.
    IOSB = IO_STATUS             ; Status return
  BSBW ERROR                     ; Check for error
```

Return Status

SS$_NORMAL	Service completed successfully.

$QIO(IO$_ACCESS)

$QIO(IO$_ACCESS) — Set Up a Virtual Circuit

Purpose

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 *VSI 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.

PSI$C_NCB Item Code	**Meaning**	**Notes**
	Mandatory items	
For PVCs, you must specify:		
PVCNAM	PVC identifier	2, 3
	Optional items	
CALLED_ EXTENSION	Called address extension	5
CALLING_ EXTENSION	Calling address extension	5
CHARGING_INFO	Charging information request	
CUG	(Bilateral) Closed User Group	1
CUM_TRST_DLY	Cumulative transit delay	
DTECLASS	Name of the DTE Class from which a member DTE will be used to make the call	3, 4
ETE_TRST_DLY	End–to–end transit delay	
EXPEDITE	Negotiate use of interrupts	
FSEL	Fast select (no restriction)	
FSEL_RES	Fast select (restricted response)	
LOCFAC	Local PSDN facilities	
LOCSUBADR (OpenVMS VAX)	Local subaddress	
MAX_TRST_DLY	Maximum acceptable transit delay	

PSI$C_NCB Item Code	**Meaning**	**Notes**
MIN_THRUCLS	Minimum throughput class (data rate)	
NET_USER_ID	Network user identifier	
PKTSIZE	Packet size	
RCV_QUOTA	Maximum receive buffer bytes	3
REMDTE	Remote DTE address	1
REVCHG	Reverse charging request	
RPOA	Remote Port of Access	
TEMPLATE	Name of template created by network management containing specified parameters	
THRUCLS	Throughput class (maximum data rate)	
TRANSIT_DELAY	Maximum transit delay	
USERDATA	User data field	6
WINSIZE	Window size	
	Ignored items	
NULL	Null item identifier	

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
  BSBW 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$_FILALRACC	For SVCs: Invalid unit number for SVC already in use by another process. For PVCs (X.25 only): The PVC is already being used by another process.
SS$_IVBUFLEN	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$_IVDEVNAM	The format of the NCB is invalid, or an error was detected while processing the NCB. Check the secondary status in the IOSB. Details of the secondary status are given below.
SS$_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.

For OpenVMS VAX, 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_PKTBAD	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_WORDBAD	The value of one of the transit delay items has been reduced to 65,535

If the first word of the IOSB contains SS$_ABORT or SS$_IVDEVNAM, the third word can contain one of the status values shown in the following table.

Code	**Meaning**
PSI$C_ERR_BADNAME	Bad counted string parameter. Correct and retry.
PSI$C_ERR_BADPARM	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.
PSI$C_ERR_CONFLICT	Conflicting items specified.
PSI$C_ERR_FACLNG	Facilities too long.
PSI $C_ERR_DTENOTAVAILABLE	The requested DTE is not available.
PSI $C_ERR_DTENOTINCLASS	The requested DTE is not a member of the specified DTE class.
PSI $C_ERR_DTENOTINGROUP	The requested DTE is not a member of the specified group.
PSI$C_ERR_INVEXP	Invalid use of expedited data negotiation.
PSI$C_ERR_INVITEM	Invalid item code.
PSI$C_ERR_INVNUM	Invalid ASCII number.
PSI$C_ERR_INVTRSTDLY	Invalid use of end–to–end transit delay facility; for example, MAX_TRST_DLY without ETE_TRST_DLY, or ETE_TRST_DLY without CUM_TRST_DLY.
PSI$C_ERR_L3ERR	Error returned from level 3.
PSI$C_ERR_MANYICI	More than one internal call identifier given.
PSI$C_ERR_NOACCESS	The X.25 Access module has been disabled or deleted.
PSI$C_ERR_NODTES	No DTE is available on which to make the call.

Code	Meaning
PSI$C_ERR_NOICI	No internal identifier specified.
PSI$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.
PSI$C_ERR_NOLOCAL	The ACP needs more logical workspace memory. Increase the nonpaged pool and retry.
PSI$C_ERR_NONONPAG	Not enough local workspace memory. Increase the nonpaged pool and retry.
PSI$C_ERR_NOSUCHDTE	The specified DTE is not known.
PSI $C_ERR_NOSUCHDTECLASS	The specified DTE class is not known.
PSI$C_ERR_NOSUCHGROUP	The specified group is not known.
PSI$C_ERR_NOSUCHPVC	The specified PVC is not known (X.25 only).
PSI $C_ERR_NOSUCHSECURITY DTECLS	The security DTECLASS has not been found.
PSI $C_ERR_NOSUCHTEMPLATE	The specified template is not known.
PSI$C_ERR_NOTIMP	The requested feature is not yet implemented.
PSI$C_ERR_NOTRANS	No translation for this name (for example, unknown user group).
PSI$C_ERR_PVCALRACC	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.
PSI$C_ERR_RECURLMT	Recursion limit reached, contact your local HP support representative for information about the variety of service options available to you and the procedures for submitting software problem reports.
PSI$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.

$QIO(IO$_ACCESS!IO$M_ABORT)

$QIO(IO$_ACCESS!IO$M_ABORT) — Reject a Call

Purpose

$QIO(IO$_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.

PSI$C_NCB Item Code	Meaning	Notes
Mandatory items		
ICI	Incoming call identifier	
Optional items		
CALLED_ EXTENSION	Called address extension	
CAUSE	Code for PSDN clearing a call	2
DIAGCODE	Diagnostic code	
LOCFACR	Local PSDN facilities	
RESPDATA	Fast select response data	3
Ignored items		
ADDR_MOD_RSN	Reason for modifying line address	
CALLING_ EXTENSION	Calling address extension	
CALL_REDIR_ORIG	Original DTE filter	
CALL_REDIR_RSN	Call redirection reason	
CUG	(Bilateral) Closed User Group	
CUM_TRST_DLY	Cumulative transit delay	
DTECLASS	Name of the DTE Class from which a member DTE is used to make the call	
ETE_TRST_DLY	End–to–end transit delay	
EXPEDITE	Negotiate use of interrupts	

PSI$C_NCB Item Code	Meaning	Notes
FLT_PRI	Filter priority	
FSEL	Fast select (no restriction)	
FSEL_RES	Fast select (restricted response)	
LOCFAC	Local PSDN facilities	
LOCSUBADR (OpenVMS VAX)	Local subaddress	
MAX_TRST_DLY	Maximum acceptable transit delay	
MIN_THRUCLS	Minimum throughput class (data rate)	
NET_USER_ID	Network user identifier	
NULL	Null item identifier	
PKTSIZE	Packet size	
RCV_QUOTA	Maximum number of receive buffers	
REMDTE	Remote DTE address	
REMSUBADR	Remote DTE subaddress	
REVCHG	Reverse charging request	
RPOA	Remote Port Of Access	
TEMPLATE	Name of template created by network management containing specified parameters	
THRUCLS	Throughput class (maximum data rate)	
TRANSIT_DELAY	Maximum transit delay	
USERDATA	User data field	
WINSIZE	Window size	

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.

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.

3. Use this field only for fast select calls.

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.

$QIO(IO$_ACCESS!IO$M_ACCEPT)

$QIO(IO$_ACCESS!IO$M_ACCEPT) — Accept a Call

Purpose

$QIO(IO$_ACCESS!IO$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 *VSI 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$_ACCESS!IO$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.

PSI$C_NCB Item Code	**Meaning**	**Notes**
	Mandatory items	
ICI	Incoming call identifier	
	Optional items	
CALLED_ EXTENSION	Called address extension	1
CUM_TRST_DLY_R	Cumulative transit delay	
EXPEDITE	Negotiate use of interrupts	
LOCFACR	Local PSDN facilities	
NET_USER_ID	Network user identifier	
RESPDATA	Fast select response data	2
PKTSIZE	Packet size	
RCV_QUOTA	Maximum number of receive buffers	
TEMPLATE	Name of template created by network management containing specified parameters	3
THRUCLS	Throughput class (maximum data rate)	
WINSIZE	Window size	
	Ignored items	
ADDR_MOD_RSN	Reason for modifying line address	
CALLING_ EXTENSION	Calling address extension	

PSI$C_NCB Item Code	Meaning	Notes
CALL_REDIR_ORIG	Original DTE filter	
CALL_REDIR_RSN	Call redirection reason	
CUG	(Bilateral) Closed User Group	
CUM_TRST_DLY	Cumulative transit delay	
ETE_TRST_DLY	End–to-end transit delay	
FSEL	Fast select (no restriction)	
FSEL_RES	Fast select (restricted response)	
LOCFAC	Local PSDN facilities	
LOCSUBADR (OpenVMS VAX)	Local subaddress	
MAX_TRST_DLY	Maximum acceptable transit delay	
MIN_THRUCLS	Minimum throughput class (data rate)	
DTECLASS	Name of a DTE Class from which a member DTE is used to make the call	
NULL	Null item identifier	
FLT_PRI	Filter priority	
REMDTE	Remote DTE address	
REMSUBADR	Remote DTE subaddress	
REVCHG	Reverse charging request	
RPOA	Remote Port Of Access	
TRANSIT_DELAY	Maximum transit delay	
USERDATA	User data field	

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:
;-
ACCEPT_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

SS$_NORMAL	Service queued successfully (R0). The virtual circuit has been accepted (IOSB).
SS$_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$_DEVOFFLINE	Local DTE or X.25 Connector node is being shut down.
SS$_FILALRACC	Invalid unit number for SVC already in use by another process.
SS$_FILNOTACC	No SVC on the specified channel to NV (X.29 only).
SS$_IVDEVNAM	The format of the NCB is invalid, or an error was detected while processing the NCB. Check the secondary status in the IOSB.
SS$_NOSUCHNODE	The incoming call you are trying to accept no longer exists; or, the incoming call identifier (PSI$C_NCB_ICI) was incorrect; or, the call is using the facility for fast select (restricted response).
SS$_OPINCOMPL	A previous call is still in progress on this channel.

$QIO(IO$_ACCESS!IO$M_REDIRECT)

$QIO(IO$_ACCESS!IO$M_REDIRECT) — Redirect a Call

Purpose

$QIO(IO$_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

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

Arguments

func	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.

PSI$C_NCB Item Code	Meaning
Mandatory items	
ICI	Incoming call identifier
In addition to this item code, **at least one** of the following item codes must be specified:	
FILTER	Name of filter to which to redirect the call. Note that the filter to which you redirect the call is not used to rematch the call parameters.
FLT_REDPRI	Redirection priority.
Optional items	
CALLED_ EXTENSION	Called address extension
CALL_REDIR_ORIG	Original DTE filter
CALL_REDIR_RSN	Call redirection reason
LOCSUBADR (OpenVMS VAX)	Local subaddress
TEMPLATE	Name of template created by network management with specified parameters
USERDATA	User data field
Ignored items	
ADDR_MOD_RSN	Reason for modifying line address
CALLING_ EXTENSION	Calling address extension
CUG	(Bilateral) Closed User Group
CUM_TRST_DLY	Cumulative transit delay
ETE_TRST_DLY	End–to–end transit delay
EXPEDITE	Negotiate use of interrupts
FSEL	Fast select (no restriction)
FSEL_RES	Fast select (restricted response)
LOCFAC	Local PSDN facilities
MAX_TRST_DLY	Maximum acceptable transit delay
MIN_THRUCLS	Minimum throughput class (data rate)
DTECLASS	Name of a DTE Class from which a member DTE is used to make the call

PSI$C_NCB Item Code	**Meaning**
NULL	Null item identifier
PKTSIZE	Packet size
FLT_PRI	Filter priority
RCV_QUOTA	Maximum number of receive buffers
REMDTE	Remote DTE address
REMSUBADR	Remote DTE subaddress
REVCHG	Reverse charging request
RPOA	Remote Port Of Access
THRUCLS	Throughput class (maximum data rate)
TRANSIT_DELAY	Maximum transit delay
WINSIZE	Window size

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`.

```
; 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:
$QIOW_S -                                   ; Issue QIO and wait
        CHAN = PSI_CHAN,-                   ; to the network device
        FUNC = #IO$_ACCESS!IO$M_REDIRECT,-  ; Function is
                                            ; redirect call
        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$_NOSUCHNODE	Either the incoming call you are trying to redirect no longer exists, or the incoming call identifier (PSI$C_NCB_ICI) is incorrect.
SS$_OPINCOMPL	A previous call is still in progress on this channel.
SS$_TOOMANYREDS (OpenVMS VAX)	This call has been redirected more than eight times, and cannot be redirected again.

$QIO(IO$_ACPCONTROL)

$QIO(IO$_ACPCONTROL) — Declare a Network Process

Purpose

$QIO(IO$_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:

  ```
  X25L3-pid-xxxx
  ```

  where:

  `pid` is the process identifier (in hex)

  `xxxx` is a unique number (in hex)

- 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:

```
X29-pid-xxxx
```

where:

`pid` is the process identifier (in hex)

`xxxx` is a unique number (in hex)

Further filters can be added by issuing more than one QIO with a function code of IO $_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 $_ACPCONTROL) was issued.

For information about displaying the filters in the X25 Access module, see *VSI X.25 for OpenVMS Management Guide* [https://docs.vmssoftware.com/vsi-x-25-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

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

Arguments

func	IO$_ACPCONTROL

p1	Address of the quadword descriptor of a 5–byte block. The format of the 5–byte block is:
	`.BYTE NFB$C_DECLNAME`
	`.LONG 0`
	The contents of the 5–byte block are:
	A function type code (one byte)
	A longword parameter value
	The function type is a symbol defined by the $NFBDEF macro in the library `LIB.MLB`.

p2	Starting virtual address of the quadword descriptor of a network process declaration block (see below).
p3	Not used.
p4	Not used.
p5	Not used.
p6	Not used.

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$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

PSI$C_NTD Item Code	**Meaning**	**Content**
	Mandatory item codes	
ACCLVL	Access level	Counted ASCII string, which declares the filter of either: ● An X25 listener (ACCLVL contains the string "X25L3") ● An X29 listener (ACCLVL contains the string "X29") See Note 1.
	Optional item codes	
CALLED_DTE	Called DTE address	Counted string, containing the address of the DTE originally called. This item is used after a call has been redirected.
DATMSK	Data mask	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 PSI $C_NTD_USRDATA. If PSI$C_NTD_DATMSK is not specified, no comparison is made. See Note 2.
EXTMSK	Called extension	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_NTD_EXTVAL. If

PSI$C_NTD Item Code	Meaning	Content
		PSI$C_NTD_EXTMSK is not specified, no comparison is made. See Note 3.
EXTVAL	Called extension	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$C_NTD_EXTMSK. If PSI$C_NTD_EXTVAL is not specified, no comparison is made.
INCDTE	Incoming DTE	Counted ASCII string, containing the Called DTE field from an incoming call packet.
FILTER	Name of the filter in the X25 Access module	Counted ASCII string, specifying the name of: ● 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 (OpenVMS VAX)	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 (OpenVMS VAX)	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

PSI$C_NTD Item Code	Meaning	Content
		incoming data masked by PSI$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:

Symbolic Value	Meaning
PSI$C_REDRSN_BUSY	Busy
PSI$C_REDRSN_OUT_OF_ORDER	Out of order
PSI$C_REDRSN_SYSTEMATIC	Automatic redirection

5. For OpenVMS VAX, 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.

   For OpenVMS IA-64 and Alpha, 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:                            ;Function block descriptor
            .LONG   5                   ;Length
            .ADDRESS FUNC_BUF           ;Address
  FUNC_BUF:                             ;Function block
            .BYTE NFB$C_DECLNAME
            .LONG   0
  DEC_DESC:                             ;Declaration descriptor
            .LONG   DEC_BUF_LEN         ;Length
            .ADDRESS DEC_BUF            ;Address
  DEC_BUF:                              ;Declaration block
  ACCLVL:
            .WORD   ACCLVL_LEN          ;Length field
            .WORD   PSI$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:
$QIOW_S -                        ; Issue QIO and wait
        CHAN = PSI_CHAN,-        ; to the network device
        FUNC = #IO$_ACPCONTROL-  ; Function is ACP control
        IOSB = IO_STATUS,-       ; I/O status block
        P1 = FUNC_DESC,-         ; Function descriptor address
        P2 = #DEC_DESC           ; Process declaration
                                 ; descriptor 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$C_DECLNAME         ; function type
  .LONG  0                      ; parameter
NPDB_DESC:                      ; Network Process Declaration
                                ; Buffer descriptor
  .LONG NPD_BUF_LEN             ; Size of buffer
  .ADDRESS NPD_BUF              ; Address of buffer
NPD_BUF:
NPD_ITEM1:
  .WORD  NPD_ITEM1_LEN          ; Length of first item
  .WORD  PSI$C_NTD_ACCLVL       ; Access Level
  .ASCIC  /X29/                 ; ASCII counted string
NPD_ITEM1_LEN = . - NPD_ITEM1   ; Calculate the size in words
NPD_BUF_LEN = . - NPD_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_NTD_FLT_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.

$QIO(IO$_DEACCESS)

$QIO(IO$_DEACCESS) — Clear a Virtual Circuit

Purpose

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 *VSI 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 ignored items are listed in the following table. Other items will generate an error if you use them.

PSI$C_NCB Item Code	Meaning	Notes
Optional items		
CAUSE	Code for PSDN clearing a call	1
DIAGCODE	Diagnostic code	
LOCFAC	Local PSDN facilities	
USERDATA	User data field	2
Ignored items		
NULL	Null item identifier	

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.

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

$QIO(IO$_READVBLK)

$QIO(IO$_READVBLK) — Receive Data

Purpose

$QIO(IO$_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

func	IO$_READVBLK
p1	Buffer address.
p2	Buffer length in bytes.
p3	Not used.
p4	Not used.
p5	Not used.
p6	Not used.

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 `READBUFSIZ`. 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
         READBUFSIZ = .-READBUF

; Using the System Service:
$QIOW_S -                      ; Issue QIO and wait
        CHAN = NW_CHAN,-       ; to NW
        FUNC = #IO$_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.

$QIO(IO$_WRITEVBLK)

$QIO(IO$_WRITEVBLK) — Transmit Data

Purpose

$QIO(IO$_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.

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.

Note

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.

Chapter 3. X.25 System Services

Table 3.1 summarizes the system services specific to X.25 programming. These services are detailed in the remainder of this chapter.

Table 3.1. System Services Specific to X.25 Programming

System Service	Description
$QIO(IO$_NETCONTROL, PSI$K_INTACK)	Confirms Receipt of an Interrupt
$QIO(IO$_NETCONTROL, PSI$K_INTERRUPT)	Transmits an Interrupt
$QIO(IO$_NETCONTROL, PSI$K_RESET)	Resets a Virtual Circuit or Confirms Receipt of a Reset
$QIO(IO$_NETCONTROL, PSI$K_RESTART)	Confirms Receipt of a Restart

$QIO(IO$_NETCONTROL, PSI$K_INTACK)

$QIO(IO$_NETCONTROL, PSI$K_INTACK) — Confirm Receipt of an Interrupt

Purpose

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`.

```
; Declaring the data:
```

```
INT_CHAN:
           .BLKW   1            ; Network channel
IO_STATUS:
           .BLKW   4            ; I/O status block
                                ; Using the System Service:
$QIOW_S -                       ; Issue QIO and wait
        CHAN = INT_CHAN,-       ; to the network
        FUNC = #IO$_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$_FILNOTACC	A virtual circuit does not exist on this channel.
SS$_MEDOFL	The PVC has been restarted. Issue a QIO system service call with a function code of IO$_NETCONTROL and a parameter of PSI $K_RESTART to confirm the restart and allow normal operation of the PVC.
SS$_NOSOLICIT	No interrupt message has been received.
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.

$QIO(IO$_NETCONTROL, PSI$K_INTERRUPT)

$QIO(IO$_NETCONTROL, PSI$K_INTERRUPT) — Transmit an Interrupt

Purpose

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

func	IO$_NETCONTROL
p1	Buffer address.
p2	Buffer length in bytes. Interrupts can be up to 32 octets long (depending on the PSDN), though most PSDNs restrict interrupts to one byte.
p3	Not used.
p4	Network control subfunction, PSI$K_INTERRUPT, for transmitting an interrupt.
p5	Not used.
p6	Not used.

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:
$QIOW_S -                        ; Issue QIO and wait
        CHAN = INT_CHAN,-        ; to the network
        FUNC = #IO$_NETCONTROL-  ; Function is network
                                 ; control
        IOSB = IO_STATUS,-       ; I/O status block
        P1 = INTBUF,-            ; Buffer address
        P2 = #INTBUFSIZ          ; and size
        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.

$QIO(IO$_NETCONTROL, PSI$K_RESET)

$QIO(IO$_NETCONTROL, PSI$K_RESET) — Reset a Virtual Circuit or Confirm the Receipt of a Reset

Purpose

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 PSI $K_RESTART to confirm the restart and allow normal operation of the PVC.
SS$_OPINCOMPL	A previous reset request is still in progress.

$QIO(IO$_NETCONTROL, PSI$K_RESTART)

$QIO(IO$_NETCONTROL, PSI$K_RESTART) — Confirm Receipt of a Restart

Purpose

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

SS$_NORMAL	Service successfully completed, restart has been confirmed.
SS$_FILNOTACC	A virtual circuit does not exist for this channel.
SS$_NOSOLICIT	No restart has been received.
SS$_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.

Chapter 4. X.29 System Services

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

System Service	Description
$QIO(IO$_NETCONTROL, PSI$K_X29_READ)	Reads X.29 Terminal Characteristics
READ Subfunctions:	
PSI$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.
$QIO(IO$_NETCONTROL, PSI $K_X29_READ_SPECIFIC)	Reads Specific X.29 Parameters
READ_SPECIFIC Subfunction:	
PSI$K_X29_PAD_PARAMS	Returns specific PAD parameters into the specified buffer
$QIO(IO$_NETCONTROL, PSI$K_X29_SET)	Sets X.29 Terminal Characteristics
SET Subfunctions:	
PSI$K_X29_BREAK_ACTION	Defines the action, or series of actions, that NV is to take when an "Indication–of–Break" X.29 message is received
PSI$K_X29_HANGUP_PARAMS	Sets the hangup PAD parameter template
PSI$K_X29_HOST_ECHO_PARAMS	Sets the host–echo PAD parameter template
PSI$K_X29_HOLD_TIMER	Sets the Hold Timer to units of 1/10 of a second
PSI$K_X29_INT_ACTION	Sets the actions to be taken when an interrupt issued at the X.29 terminal is received as an X.25 Interrupt
PSI$K_X29_LOCAL_ECHO_PARAMS	Sets the local–echo PAD parameter template
PSI$K_X29_PAD_PARAMS	Assigns values to specified PAD parameters

System Service	Description
PSI$K_X29_PAD_RESELECTION	Sends a PAD reselection message to the remote PAD
PSI$K_X29_TEMP_NOHANG	Sets the X.29 terminal to enable TEMP_NOHANG

$QIO(IO$_NETCONTROL, PSI$K_X29_READ)

$QIO(IO$_NETCONTROL, PSI$K_X29_READ) — Read X.29 Terminal Characteristics

Purpose

$QIO(IO$_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	IO$_NETCONTROL
p1	Buffer address.
p2	Buffer size in bytes. The length of the buffer depends on the parameter *p3*.
p3	One of the following Network Control Subfunctions: PSI$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$K_X29_TEMP_NOHANG These subfunctions are described in subsequent sections.
p4	PSI$K_X29_READ
p5	Not used.
p6	Unit number of the NV device.

PSI$K_X29_BREAK_ACTION

PSI$K_X29_BREAK_ACTION — READ Subfunction

Purpose

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 *VSI 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.

PSI$K_X29_HANGUP_PARAMS

PSI$K_X29_HANGUP_PARAMS — READ Subfunction

Purpose

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 *VSI 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
    P6 = NV_UNIT                ; NV unit number
  BSBW IO_ERROR       ; Check for I/O error
;+
```

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.

PSI$K_X29_HOLD_TIMER

PSI$K_X29_HOLD_TIMER — READ Subfunction

Purpose

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
         $QIOW_S -                              ; Issue QIO and wait
                 IOSB = IO_STATUS,-             ; I/O Status Block
                 CHAN = NW_CHAN,-               ; Control channel to NW
                 FUNC = #IO$_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.

PSI$K_X29_HOST_ECHO_PARAMS

PSI$K_X29_HOST_ECHO_PARAMS — READ Subfunction

Purpose

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 *VSI 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 Host-echo 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$K_X29_HOST_ECHO_PARAMS,-
    -                           ; read subfunction specifies
    -                           ; which template to manipulate
    P4 = #PSI$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.

PSI$K_X29_INT_ACTION

PSI$K_X29_INT_ACTION — READ Subfunction

Purpose

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 *VSI 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.

PSI$K_X29_LOCAL_ECHO_PARAMS

PSI$K_X29_LOCAL_ECHO_PARAMS — READ Subfunction

Purpose

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 *VSI 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
;
;-
$QIOW_S - ; Issue QIO and wait
CHAN = NW_CHAN,- ; to network device
FUNC = #IO$_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$K_X29_LOCAL_ECHO_PARAMS,-
- ; read subfunction specifies
- ; which template
P4 = #PSI$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.

PSI$K_X29_PAD_PARAMS

PSI$K_X29_PAD_PARAMS — READ Subfunction

Purpose

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 *VSI 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.

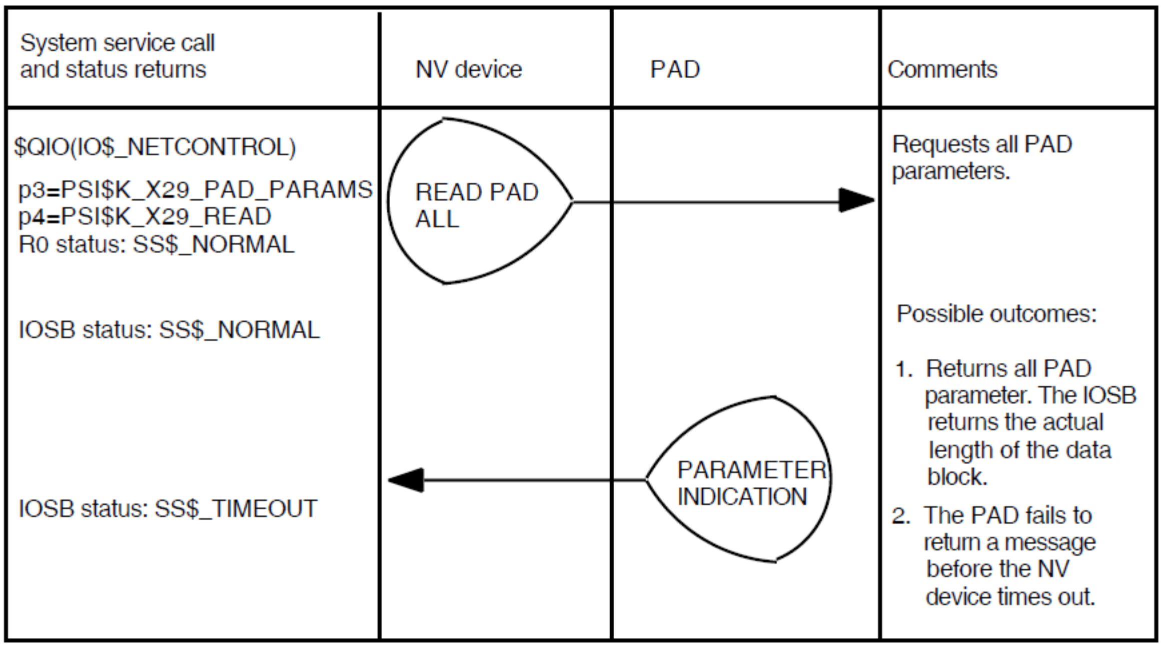
The following example and Figure 4.1 show:

- 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


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

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.

PSI$K_X29_TEMP_NOHANG

PSI$K_X29_TEMP_NOHANG — READ Subfunction

Purpose

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
;-
  $QIOW_S  -                    ; QIO and wait
    IOSB = IO_STATUS,-          ; I/O status block
    CHAN = NW_CHAN,-            ; channel to NW
    FUNC = #IO$_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.

$QIO(IO$_NETCONTROL, PSI $K_X29_READ_SPECIFIC)

$QIO(IO$_NETCONTROL, PSI$K_X29_READ_SPECIFIC) — Read Specific X.29 Parameters

Purpose

$QIO(IO$_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

func	IO$_NETCONTROL
p1	Buffer address.
p2	Buffer size in bytes. The length of the buffer depends on the parameter *p3*.
p3	The following Network Control Subfunction: PSI$K_X29_PAD_PARAMS The subfunction is described in the following section.
p4	PSI$K_X29_READ_SPECIFIC
p5	Not used.

p6	Unit number of the NV device.

PSI$K_X29_PAD_PARAMS

PSI$K_X29_PAD_PARAMS — READ_SPECIFIC Subfunction

Purpose

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.

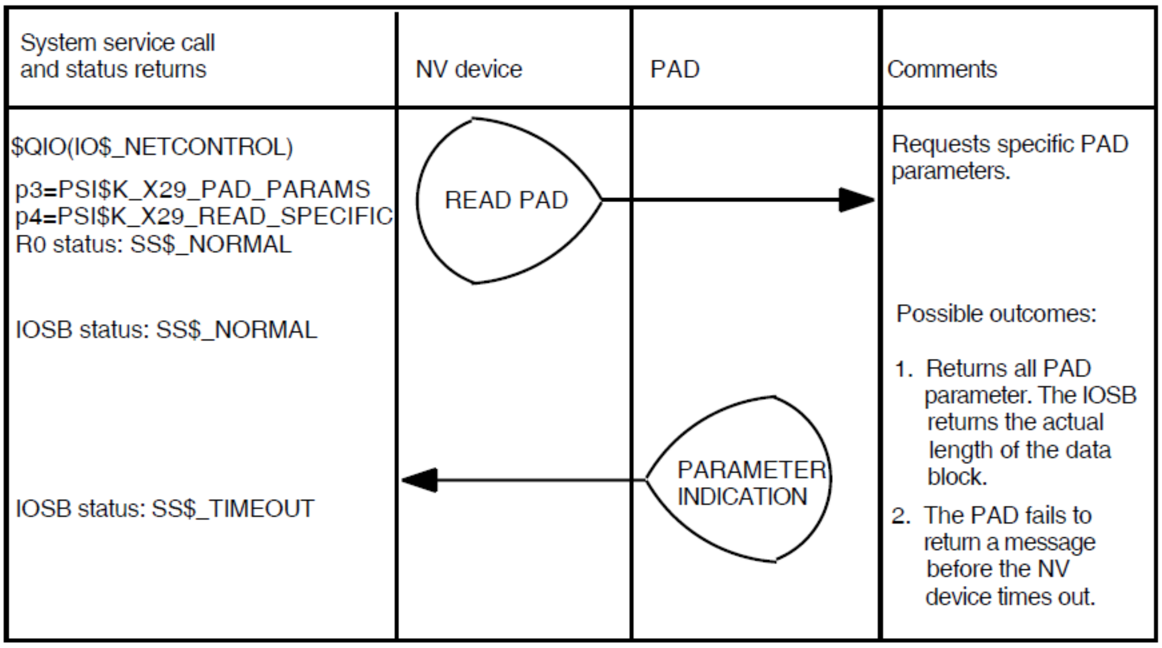
On completion of the QIO, the buffer is overwritten with the parameters returned from the PAD.

For a description of the PAD Parameter List, refer to the *VSI 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


System service call and status returns
NV device
PAD
Comments
$QIO(IO$_NETCONTROL)
p3=PSI$K_X29_PAD_PARAMS
p4=PSI$K_X29_READ_SPECIFIC
R0 status: SS$_NORMAL
IOSB status: SS$_NORMAL
IOSB status: SS$_TIMEOUT
READ PAD
PARAMETER INDICATION
Requests specific PAD parameters.
Possible outcomes:
1. Returns all PAD parameter. The IOSB returns the actual length of the data block.
2. The PAD fails to return a message before the NV device times out.

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.

$QIO(IO$_NETCONTROL, PSI$K_X29_SET)

$QIO(IO$_NETCONTROL, PSI$K_X29_SET) — Set X.29 Terminal Characteristics

Purpose

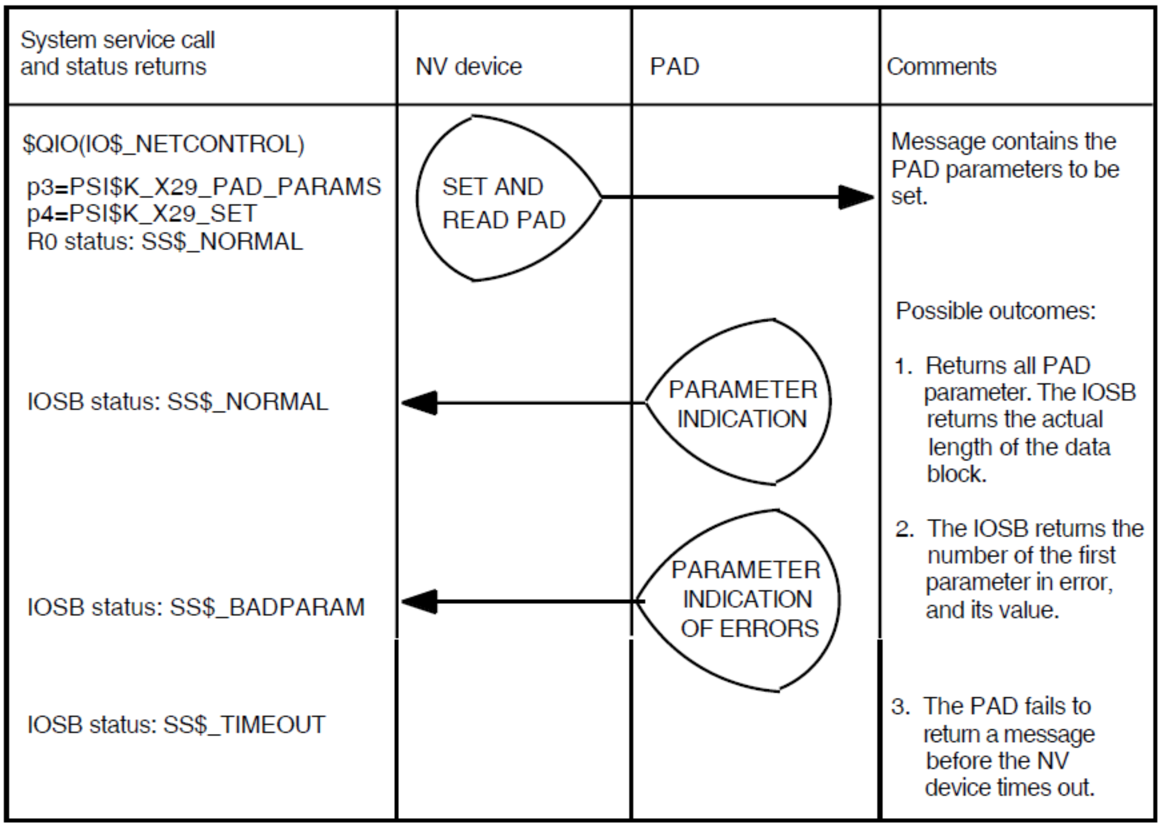
$QIO(IO$_NETCONTROL, PSI$K_X29_SET) is used in X.29 programs to set PAD parameters and NV terminal characteristics.

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


System service call and status returns
NV device
PAD
Comments
$QIO(IO$_NETCONTROL)
p3=PSI$K_X29_PAD_PARAMS
p4=PSI$K_X29_SET
R0 status: SS$_NORMAL
SET AND READ PAD
Message contains the PAD parameters to be set.
Possible outcomes:
IOSB status: SS$_NORMAL
PARAMETER INDICATION
1. Returns all PAD parameter. The IOSB returns the actual length of the data block.
IOSB status: SS$_BADPARAM
PARAMETER INDICATION OF ERRORS
2. The IOSB returns the number of the first parameter in error, and its value.
IOSB status: SS$_TIMEOUT
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:

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$K_X29_TEMP_NOHANG These subfunctions are described in subsequent sections.
p4	PSI$K_X29_SET
p5	Not used.
p6	Unit number of the NV device.

PSI$K_X29_BREAK_ACTION

PSI$K_X29_BREAK_ACTION — SET Subfunction

Purpose

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 *VSI 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:
  .LONG  PSI$M_X29_ACTION_PURGE!- ;Descriptor block
    PSI$M_X29_ACTION_RESET
;
STRING:
  .BYTE  STRING_LENGTH-1      ;Count
  .BYTE  24                   ;^X
  .BYTE  15                   ;^O
  STRING_LENGTH = .-STRING
  DESC_LENGTH   = .-DESC_BLOCK
;
  $QIOW_S -                   ; Issue QIO and wait
  IOSB = IO_STATUS            ; I/O Status block
  CHAN = NW_CHAN,-            ;Control channel to NW
  FUNC = #IO$_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
```

Return Status

SS$_NORMAL	Service completed successfully: break accepted for transmission.
SS$_ACCVIO	The NV Action Descriptor Block specified in *p1* cannot be read.

PSI$K_X29_HANGUP_PARAMS

PSI$K_X29_HANGUP_PARAMS — SET Subfunction

Purpose

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 of PAD parameters to be set*) * PSI$K_X29_PARAM_LENGTH

Refer to the *VSI 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.

```
.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
;-
  $QIOW_S -                     ; Issue QIO and wait
    CHAN = NW_CHAN,-            ; to network device
    FUNC = #IO$_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$K_X29_HANGUP_PARAMS,-
    -           ; Subfunction specifies
    -           ; which template to manipulate
    P4 = #PSI$K_X29_SET,-       ; NV set operation
    P6 = NV_UNIT                ; NV unit number
  BSBW  IO_ERROR      ; Check for I/O error
```

Return Status

SS$_NORMAL	Service completed successfully.
SS$_ACCVIO	The buffer specified in *p1* cannot be read.

PSI$K_X29_HOST_ECHO_PARAMS

PSI$K_X29_HOST_ECHO_PARAMS — SET Subfunction

Purpose

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:

(*number of PAD parameters to be set*) * PSI$K_X29_PARAM_LENGTH

Refer to the *VSI 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.

```
.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$B_X29_PARAM_VALUE
  .BYTE  0, 0, 0        ; Must be zero
.ENDM
NW_CHAN:
  .BLKW  1              ; Channel to NW
IO_STATUS:              ; I/O status block
  .BLKQ  1
NV_UNIT:
  .BLKL  1              ; NV unit number
TEMPLATE_BUFFER:
  PAD_PARAM_ITEM  -     ; Turn Echo off
    CODE=PSI$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
```

Return Status

SS$_NORMAL	Service completed successfully.
SS$_ACCVIO	The buffer specified in *p1* cannot be read.

PSI$K_X29_HOLD_TIMER

PSI$K_X29_HOLD_TIMER — SET Subfunction

Purpose

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 *VSI X.25 for OpenVMS Programming Guide*.

Example

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

```
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
;
  $QIOW_S -                            ; Issue QIO and wait
  IOSB = IO_STATUS                     ; I/O Status block
  CHAN = NW_CHAN,-                     ;Control channel to NW
  FUNC = #IO$_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 *p1* buffer cannot be read.

PSI$K_X29_INT_ACTION

PSI$K_X29_INT_ACTION — SET Subfunction

Purpose

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 *VSI 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 Ctrl/Y and 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
;
  $QIOW_S -                             ; Issue QIO and wait
  IOSB = IO_STATUS                      ; I/O Status block
  CHAN = NW_CHAN,-                      ;Control channel to NW
  FUNC = #IO$_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.

PSI$K_X29_LOCAL_ECHO_PARAMS

PSI$K_X29_LOCAL_ECHO_PARAMS — SET Subfunction

Purpose

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: (*number of PAD parameters to be set*) * PSI$K_X29_PARAM_LENGTH

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

Example

```
.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
;-
  $QIOW_S -                             ; Issue QIO and wait
    CHAN = NW_CHAN,-                    ; to network device
    FUNC = #IO$_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$K_X29_LOCAL_ECHO_PARAMS,-
    -        ; Subfunction specifies
    -        ; which template to manipulate
    P4 = #PSI$K_X29_SET,-               ; NV set operation
    P6 = NV_UNIT                        ; NV unit number
  BSBW IO_ERROR                 ; Check for I/O error
```

Return Status

SS$_NORMAL	Service completed successfully.
SS$_ACCVIO	The buffer specified in *p1* cannot be read.

PSI$K_X29_PAD_PARAMS

PSI$K_X29_PAD_PARAMS — SET Subfunction

Purpose

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*) * 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 *VSI 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
;-
  $QIOW_S -                     ; QIO and wait
    CHAN = NW_CHAN,-            ; NW channel
    IOSB = IO_STATUS,-          ; I/O status block
    FUNC = #IO$_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 *p1* 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 *p1* 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

PSI$K_X29_PAD_RESELECTION

PSI$K_X29_PAD_RESELECTION — SET Subfunction

Purpose

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.

PSI$C_NCB Item Code	**Meaning**
Mandatory items	
RESELECT_DTE	Reselection DTE address
Optional items	
LOCFAC	Local PSDN facilities

PSI$C_NCB Item Code	Meaning
USERDATA	User data field (See Note)
Ignored items	
NULL	Null item identifier

Note

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 6                 ; number of facility bytes
  .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
  BSBW  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.

PSI$K_X29_TEMP_NOHANG

PSI$K_X29_TEMP_NOHANG — SET Subfunction

Purpose

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 *VIS 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.

```
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
;-
  $QIOW_S -                       ; QIO and wait
    IOSB = IO_STATUS,-            ; I/O status block
    CHAN = NW_CHAN,-              ; channel to NW
    FUNC = #IO$_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.

Chapter 5. 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

IOSB	Contents	Meaning
Word 1	Return status	The completion status code returned by the system service call.
Word 2	Byte count	The number of bytes that have been processed. For read operations: the number of bytes read. For IO$_ACCESS operations: the number of bytes of the NCB processed successfully before an error.

IOSB	Contents	Meaning
Word 3	Secondary status	For IO$_ACCESS and IO$_DEACCESS: 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 $QIO(IO$_ACCESS) and $QIO(IO $_DEACCESS).
		For IO$_ACPCONTROL: 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 $QIO(IO$_ACPCONTROL).
		For IO$_NETCONTROL(PSI$K_X29_SET,PSI$K_ X29_PAD_PARAMS): If IOSB word 1 contains SS$_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,PSIK_X29_PAD_PARAMS) are with the service description in PSI$K_X29_PAD_PARAMS.
		For IO$_READVBLK: 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 $QIO(IO$_READVBLK).
Word 4	—	Internal information, ignore this field.

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:

Value	**Severity Level**
0	Warning
1	Success
2	Error
3	Informational
4	Severe (or fatal) error
5-7	(Reserved)

- 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.

Appendix A. Summary of X.25 System Service Calls

A.1. System Services for Setting Up and Clearing Communications

$ASSIGN *devnam*,*chan*,[*acmode*],[*mbxnam*]

Assign a channel.

$CANCEL *chan*

Clear a virtual call on a channel.

$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.

A.2. System Services for Handling Incoming Calls

$QIO [*efn*],*chan*,**IO$_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.

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

Declare a process as a network process.

A.3. System Services for Handling Control and Data Messages

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

Receive data.

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

Transmit data.

$QIO [*efn*],*chan*,**IO$_NETCONTROL**, [*iosb*],[*astadr*],[*astprm*],[*p1*],[*p2*],[*p3*], -

PSI$K_INTACK,[*p5*],[*p6*]

Confirm receipt of an interrupt.

$QIO [*efn*],*chan*,**IO$_NETCONTROL**, [*iosb*],[*astadr*],[*astprm*],[*p1*],[*p2*],[*p3*], -

PSI$K_INTERRUPT,[*p5*],[*p6*]

Transmit an interrupt.

$QIO [*efn*],*chan*,**IO$_NETCONTROL**, [*iosb*],[*astadr*],[*astprm*],[*p1*],[*p2*],[*p3*], -

PSI$K_RESET,[*p5*],[*p6*]

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

$QIO [*efn*],*chan*,**IO$_NETCONTROL**, [*iosb*],[*astadr*],[*astprm*],[*p1*],[*p2*],[*p3*],-

PSI$K_RESTART,[*p5*],[*p6*]

Confirm receipt of a restart.

$QIO [*efn*],*chan*,**IO$_NETCONTROL**, [*iosb*],[*astadr*],[*astprm*],*p1*,*p2*,*p3*, -

PSI$K_X29_READ,[*p5*],*p6*

Read X.29 terminal characteristics.

Appendix B. 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

$QIO [*efn*],*chan*,**IO$_NETCONTROL**, - [*iosb*],[*astadr*],[*astprm*],*p1*,*p2*,*p3*, **PSI$K_X29_READ**, [*p5*],*p6*

Read X.29 terminal characteristics.

$QIO [*efn*],*chan*,**IO$_NETCONTROL**, [*iosb*], - [*astadr*],[*astprm*],*p1*,*p2*,*p3*,**PSI$K_X29_SET**, [*p5*],*p6*

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.

Appendix 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 C.1. NCB Item Codes

Code	**Data (usage)**
Routing information	
CUG	(Bilateral) Closed User Group
DTECLASS	Name of the DTE Class from which a member DTE is used to make the call
FILTER	Filter entity
FLT_PRI	Destination priority (incoming)
FLT_REDPRI	Redirection priority (incoming)
GATEWAY	Gateway identifier (reserved usage)
ICI	Incoming Call Identifier
LOCDTE (OpenVMS VAX)	Local DTE address
LOCSUBADR (OpenVMS IA-64/ Alpha)	Local subaddress
REMDTE	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
REMSUBADR	Remote subaddress
TEMPLATE	Name of template created by network management, with specified parameters
Routing information: PVC only	
PASSWORD	User password
PVCNAM	PVC identifier—this must be included in the NCB for PVCs (continued on next page)
Routing information: PSDN facilities	
ADDR_MOD_RSN	Reason for modifying called line address (incoming)
CALLED_EXTENSION	Called address extension
CALLING_EXTENSION	Calling address extension
CALL_REDIR_ORIG	Original DTE destination of redirected call (incoming)
CALL_REDIR_RSN	Call redirection reason (incoming)

Code	**Data (usage)**
LOCFAC	Local PSDN facilities (outgoing)
LOCFACR	Local PSDN facilities (incoming)
NET_USER_ID	Network user identifier
RPOA	Remote Port Of Access
User data	
NULL	Null item identifier
USERDATA	User data field
User data: SVC only	
FSEL	Fast select without restriction (outgoing)
FSEL_RES	Fast select with restricted response (outgoing)
RESPDATA	Fast select response data (outgoing)
Diagnostics	
CAUSE	Code for PSDN clearing a call
DIAGCODE	Diagnostic code
REASON	Code for X.25 for OpenVMS clearing a call
Charging information	
REVCHG	Reverse charging request (outgoing)
Charging information: PSDN facilities	
CHARGE_MON	Monetary units for charging (incoming)
CHARGE_SEG	Segment count for charging (incoming)
CHARGE_TIME	Elapse time for charging (incoming)
CHARGING_INFO	Charging information request (outgoing)
Transmission facilities	
PKTSIZE	Packet size (outgoing)
THRUCLS	Throughput class (maximum data rate)
RCV_QUOTA	Total size of receive buffers (in bytes)
WINSIZE	Window size (outgoing)
Transmission parameters: PSDN facilities	
CUM_TRST_DLY	Cumulative transit delay (outgoing)
CUM_TRST_DLY_R	Cumulative transit delay (incoming)
ETE_TRST_DLY	End–to–end transit delay (outgoing)
EXPEDITE	Negotiate use of expedited data (interrupts)
MAX_TRST_DLY	Maximum acceptable transit delay
MIN_THRUCLS	Minimum throughput class (for data rate)
TRANSIT_DELAY	Requested maximum transit delay

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 $QIO(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 $QIO(IO$_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 (OpenVMS IA-64/Alpha)

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 (OpenVMS VAX)

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.

PSI$C_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 = (*packet-size* + 276)

Maximum value = (*packet-size* + 276) * *window-size*

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

Data format: longword value.

PSI$C_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 C.2. PSI$C_NCB_REASON Codes

Type Code	Content
PSI$C_L3_NETWRK	PSDN initiated
PSI$C_L3_NETERR	PSDN protocol error
PSI$C_L3_LNKDWN	Communications link failed
PSI$C_L3_LNKUP	Communications link operational
PSI$C_L3_LNKRRT	Communications link restarted
PSI$C_L3_GATDISC	Connection to Gateway (Multihost node) disconnected
PSI$C_L3_NETDISC	Connection to Gateway (Multihost node) lost
PSI$C_L3_LOCMGT	Network management function
PSI$C_L3_CALCOL	Call collision
PSI$C_L3_NETTIM	Timeout on network

PSI$C_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.

PSI$C_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.

PSI$_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.

PSI$C_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 `DATADATADATA`).

```
OPEN_INFO_START:
REMOTE_DTE:
        .WORD        REMOTE_DTE_LENGTH
        .WORD        PSI$C_NCB_REMDTE
        .ASCIC       /234219876543/
REMOTE_DTE_LENGTH =  .-REMOTE_DTE
DTECLASS:
        .WORD        DTECLASS_LENGTH
        .WORD        PSI$C_NCB_DTECLASS
        .ASCIC       /PSS/
DTECLASS_LENGTH =    .-DTECLASS
REMOTE_DTE_SUB:
        .WORD        REMOTE_DTE_SUBLEN
        .WORD        PSI$C_NCB_REMSUBADR
        .ASCIC       /26/
REMOTE_DTE_SUBLEN =  .-REMOTE_DTE_SUB
OPEN_DATA:
        .WORD        OPEN_DATA_LENGTH
        .WORD        PSI$C_NCB_USERDATA
        .ASCIC       /DATADATADATA/
OPEN_DATA_LENGTH =   .-OPEN_DATA
FAST_SELECT:
        .WORD        FAST_SELECT_LENGTH
        .WORD        PSI$C_NCB_FSEL
FAST_SELECT_LENGTH = .- FAST_SELECT
OPEN_INFO_LENGTH =   .- OPEN_INFO_START
```

Appendix D. Mailbox Messages

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.

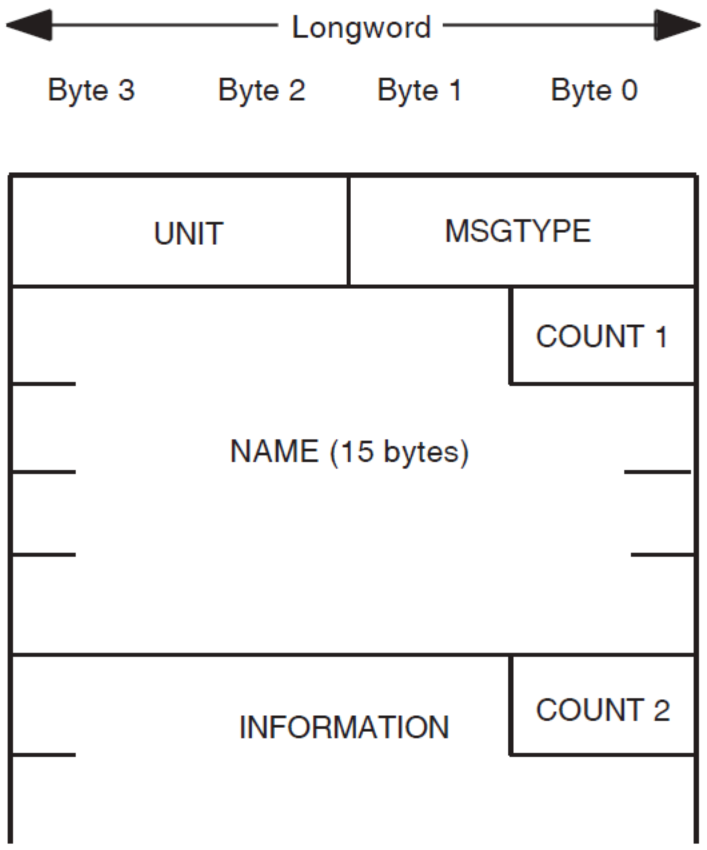
For incoming calls, your program should read the NCB in the mailbox associated with the channel that received the call. Your program must quote the incoming call identifier (PSI$C_NCB_ICI) to accept or reject an incoming call.

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.

Figure D.1. Mailbox Message Structure



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_NETWRK: Initiated by the PSDN PSI$C_L3_NETERR: PSDN protocol error PSI$C_L3_LNKDWN: Link down PSI$C_L3_LNKUP: Link up PSI$C_L3_LNKRRT: Link restarted PSI$C_L3_LOCMGT: Network 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

Type Code	Meaning	Mailbox Information
MSG$_INTMSG	Interrupt message	Interrupt byte
MSG$_CONNECT	Incoming call/Call confirm	NCB
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: ● Incoming request to clear the virtual circuit ● Completion of outgoing request to clear the virtual circuit	NCB
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

Type Code	Meaning	Mailbox Information
MSG$_CONNECT	Incoming call confirm	NCB

Type Code	Meaning	Mailbox Information
MSG$_DISCON	Incoming call reject	NCB
MSG$_TRMUNSOLIC	Incoming call	NCB

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.

On OpenVMS IA-64/Alpha systems, 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*VSI OpenVMS System Services Reference Manual: A-GETUAI* [https://docs.vmssoftware.com/vsi-openvms-system-services-reference-manual-a-getuai/#JUN_143].

Appendix E. Standard PAD Parameters

This appendix describes the standard PAD parameters. Table E.1 lists the parameter codes and parameter numbers.

Table E.1. PAD Parameter Codes

Parameter Number	Code
1	PSI$K_X29_PAR_ESCAPE
2	PSI$K_X29_PAR_ECHO
3	PSI$K_X29_PAR_FORWARD
4	PSI$K_X29_PAR_TIMEOUT
5	PSI$K_X29_PAR_HOSTSYNC
6	PSI$K_X29_PAR_MESSAGES
7	PSI$K_X29_PAR_BREAK
8	PSI$K_X29_PAR_DISCARD
9	PSI$K_X29_PAR_CRFILL
10	PSI$K_X29_PAR_WRAP
11	PSI$K_X29_PAR_SPEED
12	PSI$K_X29_PAR_TTSYNC
13	PSI$K_X29_PAR_NEW_LINE
14	PSI$K_X29_PAR_LFFILL
15	PSI$K_X29_PAR_EDIT
16	PSI$K_X29_PAR_DELETE
17	PSI$K_X29_PAR_LINE_DELETE
18	PSI$K_X29_PAR_REDISPLAY
19	PSI$K_X29_PAR_DISPLAY_EDIT
20	PSI$K_X29_PAR_RESTRICT_ECHO
21	PSI$K_X29_PAR_PARITY
22	PSI$K_X29_PAR_PAGE_WAIT

Parameter 1 (PSI$K_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 (PSI$K_X29_PAR_ECHO)

This specifies whether the PAD echoes the input entered at the X.29 terminal. Refer to the *VSI 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 (PSI$K_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 (PSI$K_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 (PSI$K_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 (PSI$K_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 (PSI$K_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:

Speed (bits/s)	**Parameter Value**	**Speed (bits/s)**	**Parameter Value**
50	10	1200	3
75	5	75/1200	11
100	9	1800	7
110	0	2400	12
134.5	1	9600	14
150	6	19200	15
200	8	48000	16
300	2	56000	17
600	4	64000	18

Parameter 12 (PSI$K_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 (PSI$K_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 to the terminal.
4	**LF** inserted after every **CR** that is echoed as data to the terminal.

Parameter 14 (PSI$K_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 (PSI$K_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 (PSI$K_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 (PSI$K_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 (PSI$K_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 (PSI$K_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 (PSI$K_X29_PAR_RESTRICT_ECHO)

This specifies which characters are not echoed. Legal values are any combination of the following:

0	All characters echoed.
1	**CR**
2	**LF**
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 PSI$K_X29_PAR_DELETE, PSI$K_X29_PAR_LINE_DELETE, PSI$K_X29_PAR_REDISPLAY_LINE).
128	**DEL**, and all other control characters.

Parameter 21 (PSI$K_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 (PSI$K_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.

Appendix F. Programming Examples

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 IA-64 and OpenVMS Alpha systems. Table F.2 summarizes the languages for which example programs are provided.

Table F.1. Programming Examples (OpenVMS IA-64 and OpenVMS Alpha)

Program	Description
`X25$CHARGING`	An example of a charging program to analyze X.25 accounting records.
`X25$RECEIVE`	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$SEND.
`X25$SEND`	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$RECEIVE.
`X25$X29_DESTINATION`	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.

Table F.2. Program/Language Matrix (OpenVMS IA-64 and OpenVMS Alpha)

Example Program	C	COBOL	FORTRAN	MACRO	PASCAL
`X25$CHARGING`	Yes	-	-	-	-
`X25$RECEIVE`	Yes	Yes	Yes	Yes	Yes
`X25$SEND`	Yes	Yes	Yes	Yes	Yes
`X25$X29_ DESTINATION`	Yes	-	-	Yes	Yes

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)

Program	Description
`PSI$CHARGING`	An example of a charging program to analyze X.25 accounting records.
`PSI$X25_RECEIVE`	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$X25_SEND.
`PSI$X25_SEND`	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$X25_RECEIVE.
`PSI$X29_DESTINATION`	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.

Program	Description
X25$X29_NETPROCESS	A simple example of a program in which an X.29 network process validates a password before allowing the X.29 caller to log in.
X25$X29_NV_UNIT_NUMBER	An example that determines the unit number of a device, given the device name.
X25$X29_OUTGOING	A simple example of an X.29 program that is run to establish an X.29 circuit from a host to a remote PAD.

Table F.4. Program/Language Matrix (OpenVMS VAX)

Example Program	BASIC	C	COBOL	FORTRAN	MACRO	PASCAL
PSI$CHARGING	-	Yes	-	-	-	-
PSI$X25_RECEIVE	Yes	Yes	Yes	Yes	Yes	Yes
PSI$X25_SEND	Yes	Yes	Yes	Yes	Yes	Yes
PSI$X29_DESTINATION	-	Yes	-	-	Yes	Yes
PSI$X29_NETPROCESS	-	-	-	-	Yes	-
PSI$X29_NV_UNIT_NUMBER	-	-	-	-	Yes	-
PSI$X29_OUTGOING	-	Yes	-	-	Yes	Yes