TOPS–10
PSI User's Guide

AA–CK81A–TB

February 1985

This manual describes how to write programs that transfer data over a Public Packet Switching Network (PPSN). The manual also describes how to gain access to a TOPS–10 host through a PPSN.

OPERATING SYSTEM: TOPS–10 V7.02

SOFTWARE: DECnet–10 V3.0
GALAXY V4.1
TOPS–10 PSI Gateway V1.0
TOPS–10 PSI Gateway Access Libraries V1.0

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

Northeast/Mid-Atlantic Region

Digital Equipment Corporation
PO Box CS2008
Nashua, New Hampshire 03061
Telephone:(603)884–6660

Central Region

Digital Equipment Corporation
Accessories and Supplies Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone:(312)640–5612

Western Region

Digital Equipment Corporation
Accessories and Supplies Center
632 Caribbean Drive
Sunnyvale, California 94086
Telephone:(408)734–4915

digital equipment corporation • marlboro, massachusetts

First Printing, February 1985

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The following are trademarks of Digital Equipment Corporation:

digital™

DEC	MASSBUS	RSX
DECmate	PDP	RT
DECsystem–10	P/OS	UNIBUS
DECSYSTEM–20	Professional	VAX
DECUS	Q–BUS	VMS
DECwriter	Rainbow	VT
DIBOL	RSTS	Work Processor

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

CONTENTS

TABLES

PREFACE

This manual contains information on:

- Writing a FORTRAN-10 or MACRO-10 program that uses the TOPS-10 X.25 software to access a PPSN (Public Packet Switching Network)
- Using the TOPS-10 X.29 software to connect a terminal - through a PPSN - to a TOPS-10 host

The manual is divided into three parts:

- Part I, the Introduction, which contains Chapter 1, a discussion of packet switching
- Part II, the X.25 Reference Guide, which contains Chapter 2, a discussion of the TOPS-10 PSI (Packetnet System Interface) Software; Chapter 3, a guide to using the FORTRAN-10-callable X.25 Gateway Access Routines; and Chapter 4, a guide to using the MACRO-10-callable X.25 Gateway Access Routines
- Part III, the X.29 Reference Guide, which contains Chapter 5, a guide to using the X.29 Software to connect a terminal through a PPSN to a remote TOPS-10 host
- Appendixes, which include sample programs, a bibliography, X.29 error messages, and a glossary

The reader of this manual should:

- Be an experienced FORTRAN-10 or MACRO-10 programmer
- Have read all user documentation published by his PPSN
- Have access to CCITT publications that describe the X.3, X.25, and X.29 recommendations (see Appendix C)

Other DIGITAL documents useful to the reader of this manual are:

TOPS-10/20 FORTRAN Language Manual. Order No. AA-N383A-TK

This manual describes the language elements of FORTRAN-10 and FORTRAN-20.

DECsystem-10 MACRO Assembler Reference Manual. Order No. AA-C780C-TB

This manual describes the language elements of the MACRO-10 Assembler for the DECsystem-10. It is helpful for understanding methods of programming in assembly language on TOPS-10.

TOPS-10 LINK Reference Manual. Order No. AA-0988D-TB

This manual describes LINK-10, the linking loader for TOPS-10.

TOPS-10 Monitor Calls Manual, Volumes 1 and 2. Order Nos. AA-0974E-TB & AA-K039B-TB

These manuals describe the functions that the monitor performs to service monitor calls for assembly language programs. Volume 1 covers facilities and functions of the monitor. Volume 2 covers monitor calls, calling sequences, symbols, and GETTAB tables.

DECnet-10 User's Guide. Order No. AA-L414A-TB

This manual contains procedures and examples for:

1. The MACRO programmer who will write network application programs

2. The nonpriviledged terminal user who will use TOPS-10 commands and the Network File Transfer (NFT) utility program in network-related tasks. This manual also describes the SET HOST command which provides login capabilities.

The following graphic conventions are used in commands and subroutine call formats in this manual:

Convention/Symbol	Meaning
UPPERCASE	Uppercase letters in a subroutine call or command string indicate user input required.
lowercase	Lowercase letters in a subroutine call or command string indicate an input variable for which you must supply a value.
Spaces	Spaces separate elements of a command. Tabs or multiple spaces can also be used. Spaces must be input where shown.
Red	Red characters indicate information that you specify in typing the command.
Black	Black characters indicate system-supplied information.
RET	This symbol indicates that you press the key labeled RETURN.
ESC	This symbol indicates that you press the key labeled ESC, ALT, or SEL.

Convention/Symbol	Meaning
[BREAK]	This symbol indicates that you press the key labeled BREAK or ATTN.
[CTRL/x]	This symbol indicates that you press the key marked CTRL while you press another key, for example, [CTRL/U] or [CTRL/V]

Unless otherwise noted, all numbers in this manual are decimal.

PART I
Introduction

CHAPTER 1

INTRODUCTION TO PACKET SWITCHING

Packet switching is the transmission of data in units called packets. One type of network that performs packet switching is a Public Packet Switching Network (PPSN). A PPSN supplies each packet it transmits with a header. This header contains control information and destination fields that enable a PPSN to deliver packets in the correct order to the correct destinations. The PPSN interleaves packets from many users over shared transmission lines and routes the packets to their destinations. The PPSN determines the route each packet takes; neither the sender nor the receiver of the packets can influence that determination.

Packet switching significantly improves the efficiency of a transmission line by sharing the line between many users, therefore, reducing the amount of time the line is idle. A PPSN can also accommodate different speeds and formats of data transfer and, through its error control mechanisms, reduce the undetected error rate to an acceptably low figure.

1.1 VIRTUAL CIRCUITS

PPSNs evolved to fill the need for a data communications service that was distinct from traditional communication circuits. Any computer or terminal can communicate with one or more computers or terminals connected to the PPSN as if they were directly connected. These computers or terminals can be of any type and manufacture and can operate at different speeds. As long as they are connected to a PPSN, they can communicate with each other. The connection between the computer and the PPSN is over a leased circuit provided by a common carrier. Terminals are connected to the PPSN over leased circuits or dial-up lines (see Figure 1-1).

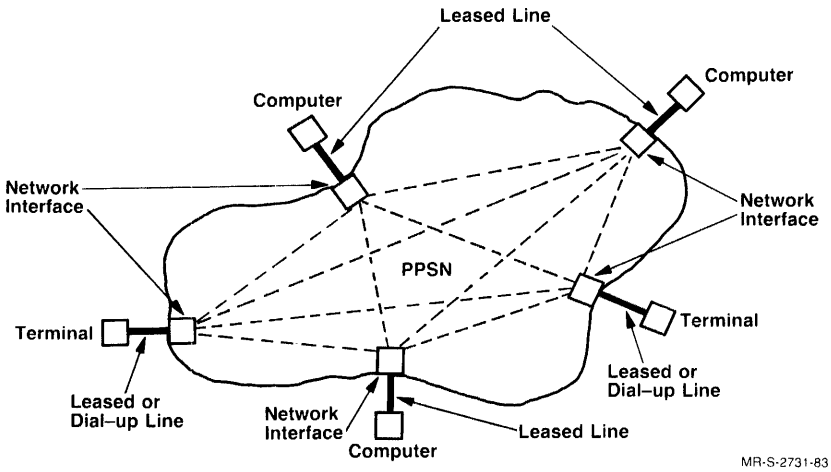


Figure 1-1: A Public Packet Switching Network (PPSN)

The techniques used within the network are transparent to both the sender and the receiver. Neither the sending computer or terminal nor the receiving computer or terminal is concerned with the control of packets in the network or with the route that packets take through the network.

The connection between the sender and receiver is called a virtual circuit and is set up by an initial exchange of packets between the sender and the receiver. Each packet contains a reference number identifying the link between the user and the network. Consequently, each virtual circuit is identified by two reference numbers: one identifying the link between the sender and the PPSN and one identifying the link between the receiver and the PPSN. These reference numbers are allocated when the sender and receiver set up a virtual circuit.

In a PPSN, such reference numbers are called Logical Channel Numbers (LCNs). The physical connection between the sender and a PPSN (and also the receiver and a PPSN) consists of logical channels. A virtual circuit associates one of the sender's logical channels with one of the receiver's logical channels, and the sender and receiver each recognize the virtual circuit only by the LCN at its own end. In general, the LCNs are different at each end of a virtual circuit.

The association between the sender and receiver can be either permanent or temporary. A permanent association is analogous to a leased circuit directly connecting the sender and the receiver. Packets bearing the appropriate LCN as transmitted by the sender are routed by the network directly to the receiver, where they are delivered bearing the LCN appropriate to that end. This association is called a Permanent Virtual Circuit (PVC).

A temporary association is analogous to a dial-up line. A temporary virtual circuit is set up only when there is data to transmit and is cleared when the transfer of this data is complete. The sender sets up such a virtual circuit by sending a Call Request packet, and the receiver accepts the circuit by sending a Call Accepted packet. Data can then be transferred between the sender and receiver. Either the sender or receiver can clear a virtual circuit by sending a Clear Request packet, at which time the other returns a Clear Confirmation packet. This type of association is called a Switched Virtual Circuit (SVC).

1.2 USER INTERFACES TO A PPSN

All user devices connected to the network fall into two classes - those that work in packet mode and those that do not. Packet-mode devices are either computers or intelligent terminals capable of conversing with the network and using packets as units of transmission. Non-packet mode devices, which include asynchronous terminals and other start-stop devices, cannot connect directly to the network but instead communicate through a packet assembly/disassembly (PAD) facility provided by the network.

To ensure that the growth of PPSNs is orderly and controlled, a special group of the CCITT (Comite Consultatif International Telegraphique et Telephonique) has established recommendations for standard network interfaces for packet-mode and non-packet-mode devices. There are four applicable recommendations:

X.25 defines the packet-mode device interface and is entitled:

- X.25 — Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode on public data networks.

X.3, X.28, and X.29 define the non-packet-mode device interface and are entitled:

- X.3 — Packet assembly/disassembly facility (PAD) in a public data network.
- X.28 — DTE/DCE interface for a start-stop mode Data Terminal Equipment accessing the packet assembly/disassembly facility (PAD) in a public data network situated in the same country.
- X.29 — Procedures for the exchange of control information and user data between a packet-mode DTE and a packet assembly/disassembly facility (PAD).

Data Terminal Equipment (DTE) is the term the CCITT uses to refer to your computer or terminal device. Data Circuit-terminating Equipment (DCE) is the term the CCITT uses to refer to the equipment of the PPSN.

1.2.1 The X.25 Recommendation

Recommendation X.25 defines only the interface between the packet-mode DTE and the DCE (see Figure 1-2) and does not state how the network functions. The sequence of events at the interface does not provide any information about what is happening at the corresponding interface between the network and the remote DTE.

The principal feature of X.25 is that it provides a full-duplex synchronous pathway that is transparent to both the sender and the receiver. The interface removes the sender's need to have precise knowledge of the characteristics of the remote DTEs. Consequently, it allows many devices to communicate over a single physical link. The interface administers all transmission by providing virtual circuits, control of packets, and error control over the network.

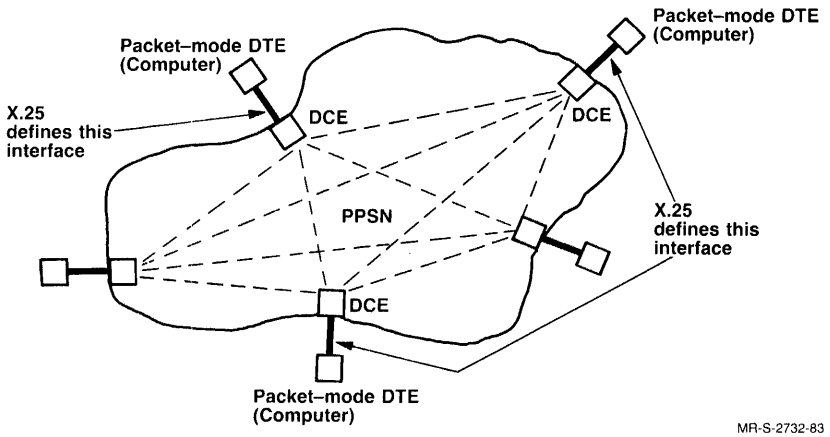


Figure 1-2: Relationship Between X.25 and the PPSN

The X.25 standard defines three levels of control procedures (or protocols) which apply to the single physical link that joins the device (the DTE) to the network interface (the DCE). See Figure 1-3.

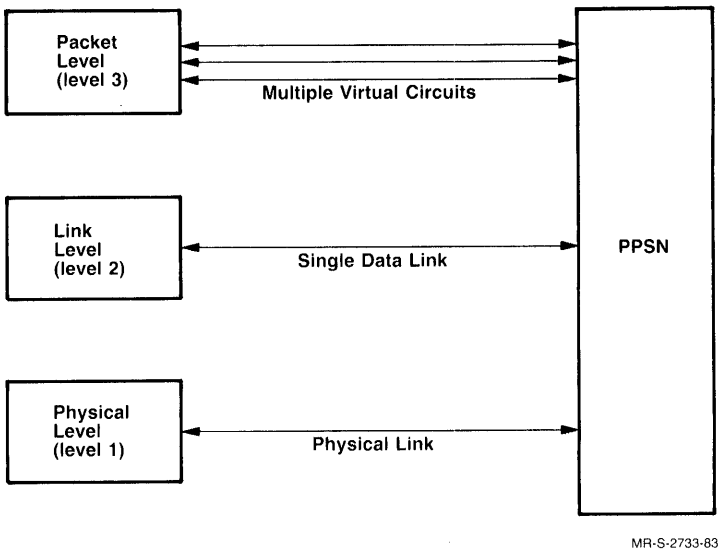


Figure 1-3: Protocol Levels

1.2.1.1 LEVEL 1 - Physical and Electrical Characteristics - Level 1 defines the physical and electrical characteristics across the DTE/DCE interface. Level 1 ensures that bits are exchanged across the interface at a fixed rate with appropriate timing information. The interface uses the X.21 bis standard. This standard is the same as specified in the EIA (Electronic Industries Association) RS232C and CCITT V.24 standards for connecting a DTE to a modem.

Level 1 is responsible for:

- Physical characteristics
- Electrical characteristics
- Fault indications

1.2.1.2 LEVEL 2 - Link Access Procedures - Level 2 defines the link access procedures across the DTE/DCE interface. The procedures are compatible with the HDLC (High-level Data Link Control) standard of ISO. The major purpose of Level 2 is to provide an error-free data communications path over a single physical link between a DTE and a DCE.

The link level is responsible for:

- Providing an efficient means of exchanging packets between the DTE and the DCE
- Providing error-detection and taking steps to recover from such errors
- Providing frame synchronization to ensure that the receiver and transmitter are in step
- Reporting procedural errors to Level 3

1.2.1.3 LEVEL 3 - Packet Procedures - Level 3 defines the packet level procedures. These procedures enable DTEs to communicate with each other across the network.

A PPSN can transmit the following types of packets:

- Call Accepted
- Call Connected
- Call Request
- Clear Confirmation
- Clear Indication
- Clear Request
- Data
- Incoming Call
- Interrupt
- Interrupt Confirmation
- Receive Not Ready
- Receive Ready
- Reset Confirmation
- Reset Indication
- Reset Request
- Restart Request
- Restart Confirmation
- Restart Indication

Prior to the transfer of data, a permanent or switched virtual circuit must be active between the two DTEs that are to communicate. Several virtual circuits can be established simultaneously with one or more users, and both PVCs and SVCs can be active at the same time. The packet level is responsible for:

- Placing the control and user data into packets
- Sequencing control of those packets
- Establishing, maintaining, and clearing virtual circuits

Figure 1-4 shows how the interface forms user data into a packet and forms the packet into a larger transmission unit called a frame.

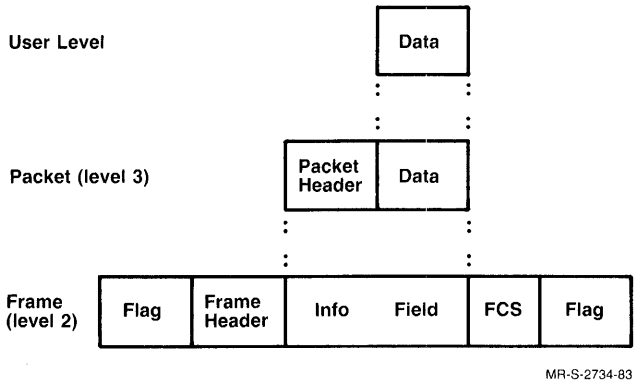


Figure 1-4: User Data/Packet/Frame Nesting

In essence, users communicate directly with Level 3 (packet procedures). The user data is divided into packet lengths and a header is added to form a packet. The coding of the data field is of no significance to the PPSN but the field itself is always restricted in length by the network (a typical length is 128 bytes). The packet header is a 24-bit field that contains the routing address, the LCN, packet identification, and optional codes (dependent on the type of packet), for example, reverse charging or priority of traffic.

The packet passes to Level 2 (link access procedures) where further control and error-detecting information and delimiting flags are added to form a frame. The control information consists of a frame identification and fields indicating the type and direction of information flow between the DTE and the DCE. The error information consists of a 16-bit Frame Checking Sequence (FCS).

Level 1 (physical and electrical characteristics) controls the physical transmission of the frame between the DTE and the DCE. The DCE checks that each frame is in the correct sequence and error free. It then passes the packet frame to the network.

Levels 1 and 2 are purely local to the DTE/DCE interface; they have no interaction with Levels 1 and 2 of the remote DTE. However, the actions taken at Level 3 do interact with the remote DTE.

1.2.2 The X.3, X.28, and X.29 Recommendations

Recommendations X.3, X.28, and X.29 specify how the assembly and disassembly of packets for start-stop devices should be provided in PPSNs.

A start-stop device needs special software to:

- Form the user data into packets
- Control the transmission and reception of those packets
- Set up virtual circuits
- Provide an internal interface to application programs

The packet assembly/disassembly (PAD) facility provides these functions. The PAD facility allows a PPSN to communicate with both packet-mode and start-stop devices.

Typically, when at a start-stop device , you first command the PAD to establish a virtual circuit to the desired remote computer. After connecting to the network, you can send commands or text as if directly connected to the remote computer.

The PAD assembles the individual characters forming the commands or text into packets and dispatches them through the network to the remote computer. Packets, arriving at the PAD from the remote computer, are disassembled and sent character-by-character to the start-stop device.

Figure 1-5 shows the relationship between X.3, X.28, X.29 and the PPSN.

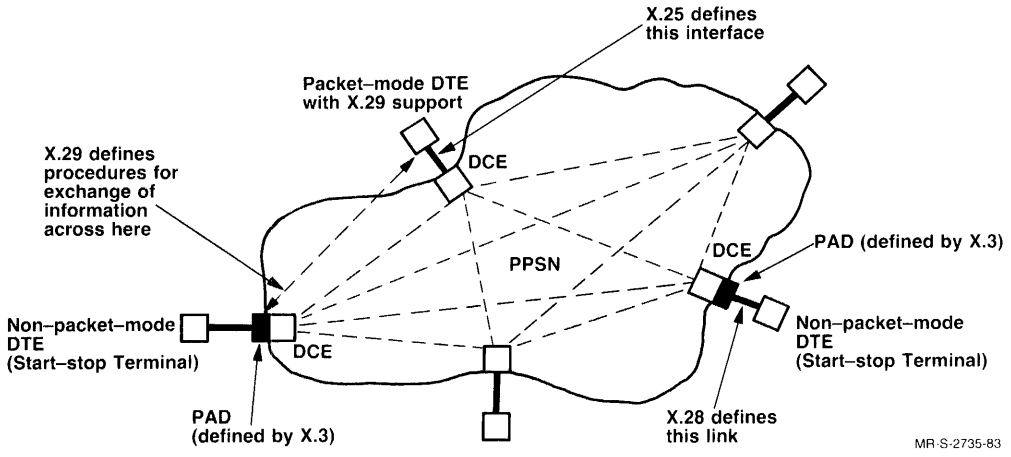


Figure 1-5: Relationship Between X.3, X.28, X.29 and the PPSN

Access from a start-stop terminal to a PAD is either by leased circuit or by dial-up line. In the latter case, a user incurs both a local telephone call charge for accessing the PAD and charges according to the tariff of the PPSN. These charges should be considerably less than those incurred by dialing and holding the equivalent long-distance telephone call.

1.3 A TYPICAL CALL

A typical call includes making a virtual circuit active, transferring data over that circuit, and clearing the circuit.

1.3.1 Setting up a Virtual Circuit

The connection between the DTE and DCE consists of a number of logical channels. The user at the local DTE sends a Call Request packet over an available logical channel to the local DCE. This Call Request packet contains the Logical Channel Number (LCN), which is selected by the DTE, and the address of the remote DTE with which the user wishes to communicate. The local DCE notes the LCN and then sends the Call Request packet to the remote DCE. The remote DCE chooses a free logical channel from those available at the remote DTE/DCE interface and sends an Incoming Call packet to the remote DTE. This Incoming Call packet contains the LCN for this remote DTE/DCE interface as well as the address of the local (calling) DTE.

If the user at the remote DTE can accept the call, he sends a Call Accepted packet to his DCE. The Call Accepted packet contains the LCN of the DTE/DCE interface at his end. His DCE sends this packet to the local DCE, which forwards it as a Call Connected packet to the local DTE.

A virtual circuit now exists between the local DTE and the remote DTE. Each DTE only knows of the circuit by the LCN at its own end.

1.3.2 Transferring Data

The two users can now transfer data between the two DTEs. Each sends data that his local interface forms into Data packets that contain the local LCN, but which arrive containing the remote LCN. The Data packets also contain packet-send and packet-receive sequence numbers. These numbers are used for flow control (see Section 1.4.2).

1.3.3 Clearing a Virtual Circuit

Either the local or the remote DTE can clear a virtual circuit by sending a Clear Request packet to the DCE. The Clear Request packet contains the LCN of the DTE/DCE interface from which it originated. The DCE forwards the Clear Request packet as a Clear Indication packet to the other DCE, which is then forwarded to the DTE. The DTE replies with a Clear Confirmation packet that is sent through the DCEs to the other DTE. The network erases the entries in the internal call control tables.

The Clear Request packet is also sent (in reply to the Call Request packet) if either of the DTEs or the network is unable to set up a virtual circuit. In these cases, the packet contains information on why the call request was refused.

A typical call sequence is shown in Figure 1-6.

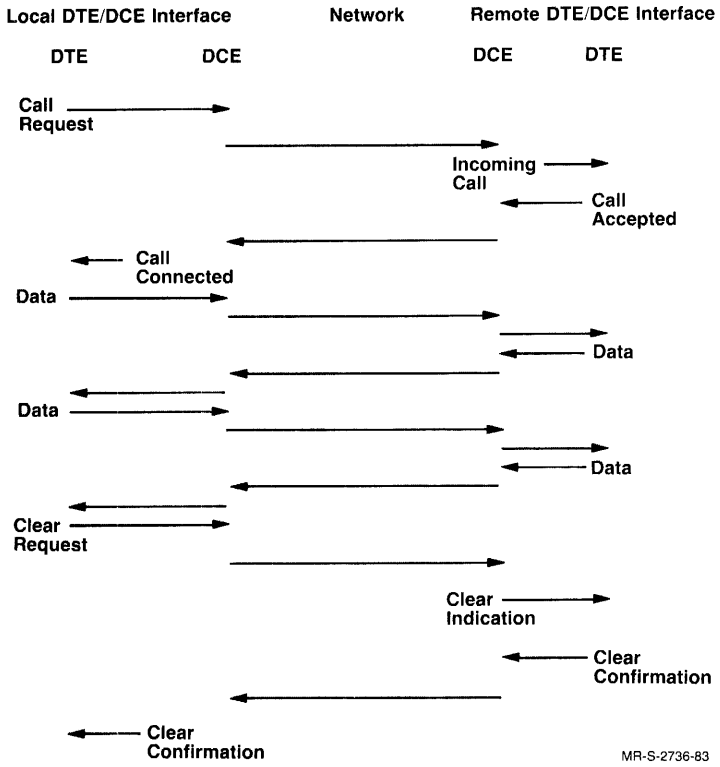


Figure 1-6: A Typical Call Sequence

1.4 FLOW CONTROL

The network provides two methods for controlling the flow of packets: one for Data packets and one for all other packets (control packets).

1.4.1 Flow Control for Control Packets

Control packets are in pairs; for example, Call Request and Call Connected packets. After a DTE sends a Call Request packet, it can do nothing until it has received a Call Connected packet. This simple method controls the flow of packets by not allowing any further packets to be sent on a particular virtual circuit until the previous action is complete. Table 1-1 lists all control packet pairs.

Table 1-1: Control Packet Pairs

Call Request	Call Connected
Clear Request	Clear Confirmation
Interrupt Request	Interrupt Confirmation
Reset Request	Reset Confirmation
Restart Request	Restart Confirmation

1.4.2 Flow Control for Data Packets

The method for controlling the flow of Data packets in Section 1.4.1 is too restrictive. Thus, a second method allows the DTE and the DCE to anticipate receipt of Data packets from the other and also allows each to send several Data packets before an acknowledgement is received. To achieve this, each Data packet contains a packet send sequence number - P(S). This number starts at zero and is increased by one for each successive packet sent on one logical channel in one direction. The interface keeps a separate count for each logical channel. You can send up to a pre-set number of packets on a logical channel in one direction before an acknowledgement must be received.

The pre-set number is called the window size and flow control is based on the concept of a window. A window is the ordered set of consecutive Data packets authorized to be transmitted across the DTE/DCE interface. The interface progressively closes the window for a logical channel as it transmits packets on that channel. The window is completely shut when the interface has transmitted a number of packets equal to the window size. At that time, the interface can send no more packets on that channel until the other end of the virtual circuit sends an acknowledgement.

That acknowledgement is the Receive Ready packet. This packet contains the LCN and a packet receive sequence number - P(R). The P(R) number indicates that all packets up to the number P(R)-1 have been received and that the next packet expected should have the number P(R). The Receive Ready packet reopens the shut window. This enables more packets, up to the window size, to be sent, starting at the one numbered P(R).

If the Receive Ready packet is sent before the window is completely shut, the window is kept partially open and Data packets can be sent continuously.

If Data packets are being sent in both directions at the same time, it is possible to cut down the number of Receive Ready packets by including the P(R) number in the Data packets being sent in the opposite direction.

1.5 ADDITIONAL PPSN FACILITIES

This section describes additional facilities that may be provided by a PPSN for controlling the exchange of packets between DTEs.

1.5.1 Interrupts

The interrupt procedure lets a DTE transmit one byte of data to a remote DTE by using the method of flow control for control packets (see Section 1.4.1). Interrupt packets are paired: an interrupt can be sent in one direction on a particular virtual circuit only if any previous interrupt has been acknowledged. The interrupt procedure has no effect on the flow of Data packets. The one byte of interrupt data is a user-defined field and is of no significance to the PPSN.

1.5.2 Resets

The reset procedure allows a DTE to reinitialize a virtual circuit by resetting the lower window edge and P(R) and P(S) numbers to zero. All Data and Interrupt packets in the network are discarded. The reset procedure uses the method of flow control for control packets (see Section 1.4.1).

1.5.3 Restarts

The restart procedure allows a DTE or DCE to clear and reinitialize all circuits. Usually, restarts are used only for non-recoverable error conditions or link initialization. During a restart, the DTE discards all outstanding Data and Interrupt packets on all channels. The DTE or DCE executes this procedure as needed; the user has no control over the procedure. For flow control, the restart procedure uses the method for control packets (see Section 1.4.1).

1.5.4 Optional Facilities

Three types of optional facilities are offered by PPSNs:

- Those that require no prior arrangement at subscription time but can be requested on a per call basis; that is, they can be requested by specifying fields in the Call Request packet.
- Those that require prior arrangement at subscription time and always apply from then on.
- Those that require prior arrangement at subscription time and then may be optionally requested on a per call basis.

The optional facilities are:

- **Bilateral Closed User Group (BCUG) -** The BCUG restricts a DTE to communicating with another single DTE. If used, this must be subscribed to with the PPSN and, when used, specified on a per call basis. The facility is applicable to SVCs only.

  There is an addition to this basic facility: BCUG with outgoing access. This allows a DTE to initiate calls to DTEs outside their BCUG.

- **Closed User Group (CUG) -** The CUG restricts a DTE to communicating with two or more DTEs in the same group. If used, this facility must be subscribed to with the PPSN and, when used, specified on a per call basis. The facility is applicable to SVCs only.

  Additions to this basic facility, which may be subscribed to separately or together are:

  -- CUG with outgoing access - allows a DTE to initiate calls to DTEs outside their CUG.

  -- CUG with incoming access - allows a DTE to receive calls from DTEs outside their CUG.

- **Throughput Class Negotiation -** Throughput class is the maximum rate of data transmission for a particular SVC. Throughput class negotiation allows a DTE to request a higher or lower data rate depending on the throughput of data packets. If used, this facility must be subscribed to with the PPSN and specified on a per call basis.

- **Default Throughput Class Assignment -** This permits a DTE to specify a default Throughput Class different from the PPSN's default. If used, this facility must be subscribed to with the PPSN.

- **Fast Select -** This allows a DTE to include a user data field in any Call Request, Call Accepted, or Clear Request packet that is longer than 16 bytes. If used, this facility must be subscribed to with the PPSN and, when used, specified on a per call basis. The facility is applicable to SVCs only.

  There are qualifiers to this basic facility, as follows:

  -- No restriction on response - allows the remote DTE to accept or reject the request to set up a virtual circuit and also reply with user data.

  -- Restriction on response - prevents the remote DTE from accepting the request to set up a virtual circuit but allows user data to be sent with the rejection.

- **Fast Select Acceptance -** This allows a DTE to accept Incoming Call packets that contain a user data field. If used, this facility must be subscribed to with the PPSN.

- **Flow Control Parameter Negotiation** - This permits selection of maximum packet sizes and window sizes in each direction of a particular virtual circuit. If used, this facility must be subscribed to with the PPSN and, when used, must be specified on a per call basis. The values specified must be acceptable to the PPSN to which the DTE is connected.

- **Nonstandard Default Packet size** - This permits a DTE to specify a default packet size different from the PPSN's default. If used, this facility must be subscribed to with the PPSN.

- **Nonstandard Default Window size** - This permits a DTE to specify a default window size different from the PPSN's default. If used, this facility must be subscribed to with the PPSN.

- **One-way Logical Channel Incoming** - This prevents a particular logical channel from handling outgoing calls. If used, this facility must be subscribed to with the PPSN.

- **One-way Logical Channel Outgoing** - This prevents a particular logical channel from handling incoming calls. If used, this facility must be subscribed to with the PPSN.

- **Incoming Calls Barred** - This prevents a DTE from receiving any calls. If used, this facility must be subscribed to with the PPSN.

- **Outgoing Calls Barred** - This prevents a DTE from initiating any calls. If used, this facility must be subscribed to with the PPSN.

- **Incoming Calls Barred Within a CUG** - This prevents a DTE from receiving any calls from within a CUG. If used, this facility must be subscribed to with the PPSN.

- **Outgoing Calls Barred Within a CUG** - This prevents a DTE from initiating any calls within a CUG. If used, this facility must be subscribed to with the PPSN.

- **Reverse Charging** - This allows a DTE to request that the remote DTE pays for the call. If used, this facility must be subscribed to with the PPSN and, when used, specified on a per call basis.

- **Reverse Charging Acceptance** - This allows a DTE to accept a request for reverse charging. If used, this facility must be subscribed to with the PPSN.

- **Extended Packet Sequence Numbering** - This allows a DTE to change the packet sequence P(S) numbering to modulo 128 instead of modulo 8. If used, this facility must be subscribed to with the PPSN.

- **Packet Retransmission** - This allows a DTE to request retransmission of one or more Data packets by specifying a P(R) in a Reject packet. If used, this facility must be subscribed to with the PPSN.

- **RPOA Selection** - This allows a DTE to specify a Remote Port of Access to another PPSN if access to more than one PPSN is permitted. If used, this facility must be subscribed to with the PPSN and, when used, must be specified on a per call basis.

PART II
X.25 Reference Guide

CHAPTER 2

THE TOPS-10 X.25 SOFTWARE

The TOPS-10 PSI software contains the following three modules:

- TOPS-10 PSI Gateway Access Routines
- TOPS-10 PSI Gateway Software
- X29SRV, the TOPS-10 X.29 Software

Figure 2-1 illustrates the placement of the TOPS-10 PSI Gateway Access Routines and the TOPS-10 Gateway Software in the central (TOPS-10) processor and the communications processor (DTE). (For information on the TOPS-10 X.29 Software, see Part III of this manual.) To communicate through a PPSN, you place TOPS-10 PSI subroutine calls within your FORTRAN-10 or MACRO-10 program. These subroutine calls are described in Chapters 3 and 4. The FORTRAN-10 or MACRO-10 program collects the data to be sent and passes it to the TOPS-10 PSI Gateway Access Routines. These routines assemble an access protocol message, set up a DECnet logical link, and pass the data to the TOPS-10 PSI Gateway Software. This software reconstitutes the message for transmission on a virtual circuit and reformulates the message into X.25 packets.

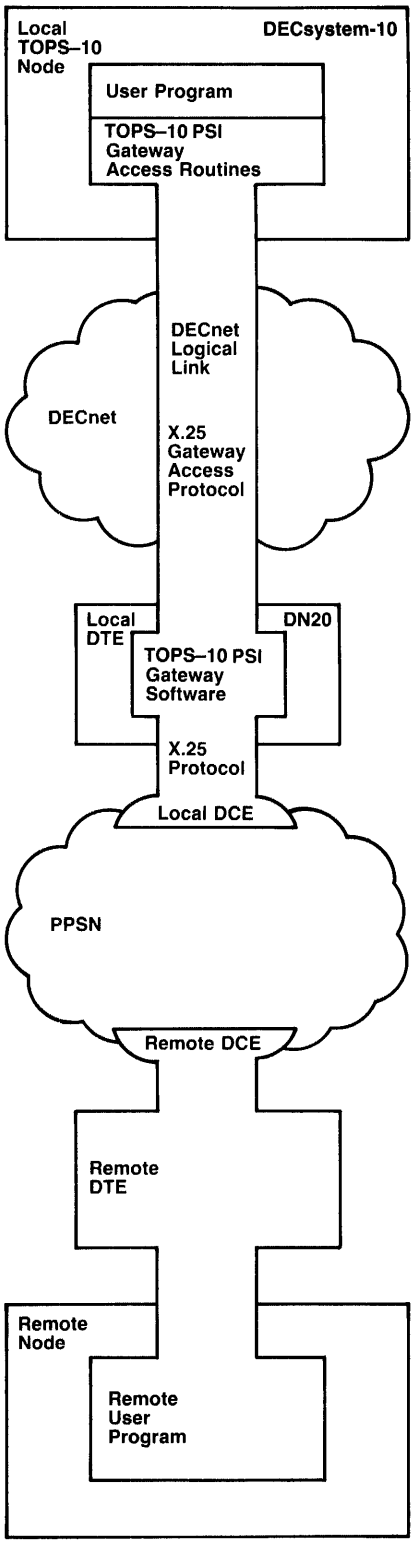


Figure 2-1: TOPS-10 PSI Gateway Software Components

THE TOPS-10 X.25 SOFTWARE

The TOPS-10 PSI Gateway Software controls data transmission, ensures error-free data, and manages the X.25 port through which communication to the PPSN occurs. This software provides the functions required to implement TOPS-10 PSI protocol levels 1, 2, and 3 (see Section 1.2.1). Once the TOPS-10 PSI Gateway Software sends the packets across the virtual circuit into the X.25 PPSN, the PPSN controls the transmission of the packets until they reach their destination. If the message must travel across more than one PPSN, each PPSN provides the facility for connecting to the adjoining PPSN. An incoming message is handled in the reverse order from an outgoing message.

2.1 TOPS-10 PSI SOFTWARE FUNCTIONS

Where there is communication between user processes at peer levels (equivalent network layers) in widely separated processors, the processes use a protocol. Where adjacent network layers communicate, they use an interface. The TOPS-10 PSI Software provides the protocols and any other services needed for you to communicate with the PPSN.

The TOPS-10 PSI Gateway Access Routines provide the interface between the X.25 user program and the local DTE, which is connected to the PPSN. This interface uses UUO calls to interact with the TOPS-10 operating system at the user level.

The TOPS-10 PSI Gateway Access Routines establish a communications path to the local DTE through a DECnet logical link to the DN20 (PSI Gateway) over which access protocol messages pass. The user program controls the logical link with subroutines that format and interpret protocol messages. The local DTE, which is connected to the PPSN, communicates with the PPSN through a virtual circuit.

Table 2-1 summarizes the subroutines available to communicate through a PPSN. Chapters 3 and 4 describe them in greater detail. Table 2-1 lists the subroutines in the order in which they should be used; subroutines that open virtual circuits are listed first; subroutines used with open virtual circuits are listed next; subroutines that close virtual circuits are listed last.

Table 2-1: TOPS-10 PSI Gateway Access Routine Functions

Function	Action	Type of Packet This Function Sends or Reads
OPEN PERMANENT CIRCUIT	Opens access to a PVC. Reserves the PVC for the exclusive use of the caller. Assigns a port in the X.25 Access module.	None.
WAIT INCOMING CALL	Designates a port as willing to accept incoming calls.	None.
INITIATE SWITCHED CIRCUIT	Issues a call to a remote DTE to establish an SVC. Assigns a port in the X.25 Access module.	Sends a Call Request packet.
READ ACCEPT DATA	Obtains data sent by the remote DTE when the remote DTE accepted a call.	Reads a Call connected packet.
READ INCOMING CALL	Obtains all data accompanying an incoming call.	Reads an Incoming call packet.
ACCEPT INCOMING CALL	Accepts an incoming call.	Sends a Call Accepted packet.
RESET VIRTUAL CIRCUIT	Resets (or acknowledges the reset of) a virtual circuit.	Sends a Reset Request or Reset Confirmation packet.
READ RESET DATA	Obtains data sent by the remote DTE when the remote DTE reset the virtual circuit.	Reads a Reset Indication packet.
SEND DATA MESSAGE	Transmits one packet of data over a virtual circuit.	Sends a Data packet.
READ DATA MESSAGE	Obtains a packet of data transmitted by a remote node over a virtual circuit.	Reads a Data packet.
SEND INTERRUPT MESSAGE	Transmits an interrupt byte over a virtual circuit.	Sends an Interrupt packet.

Table 2-1: TOPS-10 PSI Gateway Access Routine Functions (Cont.)

Function	Action	Type of Packet This Function Sends or Reads
READ INTERRUPT MESSAGE	Obtains an interrupt byte transmitted by a remote node over a virtual circuit.	Reads an Interrupt packet.
CONFIRM INTERRUPT MESSAGE	Confirms receipt of an interrupt byte.	Sends an Interrupt Confirmation packet.
READ PORT STATUS	Obtains the state of the X.25 access port.	None.
NO COMMUNICATION SEEN	Acknowledges failure of a PVC due to loss of communication with the PPSN.	None.
CLEAR SWITCHED CIRCUIT	Clears an SVC.	Sends a Clear Request packet.
READ CLEAR DATA	Obtains data sent by the remote DTE when that DTE cleared the SVC.	Reads a Clear Request packet.
TERMINATE PORT ACCESS	Releases all resources associated with a specific access port.	None.

You can use many TOPS-10 PSI Gateway Access Routine functions whether your connection is over an SVC or a PVC. However, some of the functions can be used only with one type of virtual circuit. Table 2-2 correlates the types of virtual circuits with the functions that can be used with them.

Table 2-2: Types of Virtual Circuits and the Functions that Can Be Used with Each

Functions that Can Be Used Only with SVCs	Functions that Can Be Used Only with PVCs	Functions that Can Be Used with SVCs and PVCs
ACCEPT INCOMING CALL	NO COMMUNICATION SEEN	CONFIRM INTERRUPT MESSAGE
CLEAR SWITCHED CIRCUIT	OPEN PERMANENT CIRCUIT	READ DATA MESSAGE
INITIATE SWITCHED CIRCUIT		READ INTERRUPT MESSAGE
READ ACCEPT DATA		READ PORT STATUS
READ CLEAR DATA		READ RESET DATA
READ INCOMING CALL		RESET VIRTUAL CIRCUIT
WAIT INCOMING CALL		SEND DATA MESSAGE
		SEND INTERRUPT MESSAGE
		TERMINATE PORT ACCESS

The TOPS-10 PSI Gateway Access Routines assign a port to a user program - providing the port resources exist - when the user program issues one of the following subroutine calls:

- OPEN PERMANENT CIRCUIT
- INITIATE SWITCHED CIRCUIT
- WAIT INCOMING CALL

The user program can then use the port number supplied by the TOPS-10 PSI Gateway Access Routines to refer to the virtual circuit created by these events.

2.2 PORT STATES

A **port state**, which is represented by a decimal value, is associated with each port. The state of a port changes because of user action or because of events perceived by the TOPS-10 PSI Gateway Access Software. Issuing certain commands can cause a port state change that is necessary for subsequent actions. For example, when you invoke the INITIATE SWITCHED CIRCUIT function to create a connection to a remote DTE, you must wait until the port state changes from CALLING to RUNNING before you can transmit data. You poll the port state with the READ PORT STATUS function to determine when the port has entered the RUNNING state. Table 2-3 lists port states.

Table 2-3: TOPS-10 PSI Gateway Port States

State	Decimal Value	Meaning
UNDEFINED	0	Occurs when the port is not assigned.
OPEN	1	Occurs when you issue an OPEN PERMANENT CIRCUIT function for a PVC but have not yet received a response from the local DTE. Success of the OPEN PERMANENT CIRCUIT changes the port state to RUNNING or UNSYNC.
CALLING	2	Occurs when the port user issues an INITIATE SWITCHED CIRCUIT. Remains until one of the following occurs: • The destination DTE accepts or rejects the call. • You issue a CLEAR SWITCHED CIRCUIT. • The call fails due to lack of resources in the X.25 gateway node. If the destination DTE accepts the call, the port enters the RUNNING state and data transmission can proceed.
LISTENING	3	Occurs when you issue a WAIT INCOMING CALL function to make the port available to receive an incoming call for a DECnet connection from the TOPS-10 PSI Gateway Software.
CALLED	4	Occurs when an incoming call is associated with the port and remains until you accept or reject the call (see READ INCOMING CALL in TOPS-10 PSI Gateway Access Routine Functions, Table 2-1).
RUNNING	5	Occurs when the associated virtual circuit is available for data transfer.
SYNC	6	Occurs when you issue a RESET VIRTUAL CIRCUIT call, but the reset has not yet been confirmed. In a SYNC state, the port is not available for data transfer. You must poll the port state and wait until it changes to RUNNING.
UNSYNC	7	Occurs when the PPSN or remote user initiates the reset of the virtual circuit, but you do not issue a RESET VIRTUAL CIRCUIT to acknowledge. The port is not available for data transfer. Before transferring more data, you must issue a RESET VIRTUAL CIRCUIT, which changes the port state from UNSYNC to RUNNING.

Table 2-3: TOPS-10 PSI Gateway Port States (Cont.)

State	Decimal Value	Meaning
CLEARING	8	Occurs when you invoke the CLEAR SWITCHED CIRCUIT call to clear the switched virtual circuit, but the public network has not yet sent confirmation.
CLEARED	9	Occurs when the switched virtual circuit has been cleared by the public network or the remote DTE. To exit from this state, you must close the port (see TERMINATE PORT ACCESS in TOPS-10 PSI Gateway Access Routine Functions, Table 2-1).
ERROR	10	Occurs when the TOPS-10 PSI Gateway Software detects a fatal error condition. Use the READ PORT STATUS function to determine the nature of the error. When the port state is ERROR, you cannot communicate over the PPSN. To clear this state, use the TERMINATE PORT ACCESS function.
NO COMMUNICATION	11	Occurs when one of the following has happened: • The PPSN has sent a Restart Indication packet for the associated DTE. • The frame level underlying the virtual circuit has failed. • An operator has set the associated DTE to the OFF State. You have not executed the NO COMMUNICATION SEEN function.

The sequence of port state changes is governed by whether the virtual circuit connection to the PPSN is permanent or switched. Certain port states such as LISTENING pertain only to switched virtual circuits. Tables 2-4 and 2-5 describe the port state changes.

Table 2-4: State Changes for PVC Ports

Old State	New State	Cause of Change
any	UNDEFINED	You issue a TERMINATE PORT ACCESS.
any state but OPEN	NO COMMUNICATION	Communication with the X.25 node or PPSN is lost.
NO COMMUNICATION	RUNNING	You execute the NO COMMUNICATION SEEN function.
OPEN	ERROR	One of the following: • Insufficient resources to maintain another port. • The requested PVC is already in use. • The requested PVC is not defined. • Communication with the X.25 node or PPSN has been lost.
OPEN	RUNNING	The requested PVC is assigned to you, and the remote user has not sent any unconfirmed resets.
OPEN	UNSYNC	The requested PVC is assigned to you, and there is an unconfirmed reset from the remote user. You should confirm the reset with a RESET VIRTUAL CIRCUIT.
RUNNING	SYNC	You issue a RESET VIRTUAL CIRCUIT but it is not confirmed by the PPSN.
RUNNING	UNSYNC	The PPSN requests a reset and waits for the local user to issue a RESET VIRTUAL CIRCUIT.
SYNC	RUNNING	The public network (PPSN) confirms an earlier RESET VIRTUAL CIRCUIT by you.
UNDEFINED	OPEN	You issue an OPEN PERMANENT CIRCUIT to communicate with a remote DTE.
UNSYNC	RUNNING	You acknowledge a previous reset request from the public network with a RESET VIRTUAL CIRCUIT.

Table 2-5: State Changes for SVC Ports

Old State	New State	Cause of Change
any	ERROR	Communication with the X.25 node or public network has been lost.
any	UNDEFINED	You issue a TERMINATE PORT ACCESS.
CALLED	CLEARED	An incoming call request is cleared by the remote user before you act.
CALLED	CLEARING	You reject an incoming call with a CLEAR SWITCHED CIRCUIT, but the remote user has not yet confirmed this rejection.
CALLED	RUNNING	You accept an incoming call with the ACCEPT INCOMING CALL function.
CALLING	CLEARED	The remote DTE rejects a call request originating when you request an INITIATE SWITCHED CIRCUIT.
CALLING	CLEARING	You issue a CLEAR SWITCHED CIRCUIT request after an INITIATE SWITCHED CIRCUIT before the remote DTE could respond to the INITIATE SWITCHED CIRCUIT.
CALLING	RUNNING	The remote DTE accepts a call request originating from an INITIATE SWITCHED CIRCUIT that you issued.
CLEARING	CLEARED	The public network confirms that you have cleared the circuit.
LISTENING	CALLED	An incoming call has arrived on a listening port.
RUNNING	CLEARED	The public network has cleared the circuit, and the local DTE has confirmed the clearing.
RUNNING	CLEARING	You have used the CLEAR SWITCHED CIRCUIT function to request clearing of the circuit.
RUNNING	SYNC	You issue a RESET VIRTUAL CIRCUIT not yet confirmed by the public network.
RUNNING	UNSYNC	The public network has issued a reset request and awaits a RESET VIRTUAL CIRCUIT from you.
SYNC	CLEARED	The public network has cleared the circuit, and the local DTE has confirmed the clearing.
SYNC	CLEARING	You have issued a CLEAR SWITCHED CIRCUIT to clear the circuit.

Table 2-5: State Changes for SVC Ports (Cont.)

Old State	New State	Cause of Change
SYNC	RUNNING	A previous request for a RESET VIRTUAL CIRCUIT from you is confirmed by the public network.
UNDEFINED	CALLING	You issue an INITIATE SWITCHED CIRCUIT to communicate with a remote DTE.
UNDEFINED	LISTENING	You issue a WAIT INCOMING CALL to wait for an incoming call.
UNSYNC	CLEARED	The public network clears the circuit and the X.25 software confirms the clear.
UNSYNC	CLEARING	You issue a CLEAR SWITCHED CIRCUIT request to clear the circuit.
UNSYNC	RUNNING	You acknowledge a previous reset request from the public network with a RESET VIRTUAL CIRCUIT.

Table 2-6 shows TOPS-10 Gateway Access Routine functions with corresponding port states. The port must be in the given state before the X.25 function can be issued. Table 2-6 lists the X.25 functions in the order in which they are used.

Table 2-6: TOPS-10 Gateway Access Routine Functions and Their Port States

Functions	Port States: UNDEFINED	OPEN	CALLING	CALLED	ERROR	RUNNING	SYNC	UNSYNC	CLEARING	CLEARED	LISTENING	NO COMMUNICATION
OPEN PERMANENT CIRCUIT	X											
INITIATE SWITCHED CIRCUIT	X											
WAIT INCOMING CALL	X											
READ ACCEPT DATA						X						
READ INCOMING CALL				X								
ACCEPT INCOMING CALL				X								
CLEAR SWITCHED CIRCUIT			X	X		X	X	X				
READ CLEAR DATA										X		
RESET VIRTUAL CIRCUIT						X		X				
READ RESET DATA								X				
SEND DATA MESSAGE						X						
READ DATA MESSAGE						X		X				
SEND INTERRUPT MESSAGE						X						
READ INTERRUPT MESSAGE						X		X				
CONFIRM INTERRUPT MESSAGE						X						
READ PORT STATUS	X	X	X	X	X	X	X	X	X	X	X	X
TERMINATE PORT ACCESS		X	X	X	X	X	X	X	X	X	X	X
NO COMMUNICATION SEEN												X

MR-S-3713-84

2.3 THE PROGRAMMING ENVIRONMENT

The packet switching technique used in the PPSNs is transparent to the user. After the user program collects data to be transmitted, the program passes that data to the TOPS-10 PSI Software, which forms the data into packets. The PPSN adds routing information to each packet and uses that information to carry the packet to its destination. At the destination, the PPSN discards routing information and converts the packets into their original form.

The TOPS-10 X.25 software uses the virtual circuit concept, which is described in Section 1.1, to establish a logical link between two stations in the network. The virtual circuits have the following characteristics:

- They allow simultaneous data exchange in two directions (full-duplex).
- They provide flow control, so that two stations operating at different speeds can work together.
- They permit multichannel access so that one processor connected to a PPSN with a single physical line can communicate with several destinations over several virtual circuits.
- They can be either permanent or switched.

2.4 TYPES OF USER PROGRAMS

When you write a program that calls TOPS-10 X.25 subroutines, that program can perform one or more of the following functions:

- Initiating an SVC Call and using the SVC
- Receiving an SVC Call and using the SVC
- Using a PVC

Furthermore, a program that performs one or more of these functions should contain three sections, which are:

- Initialization - initiating communication between the local and remote nodes
- Sending and receiving - sending and receiving data and interrupts as well as handling resets between the local and remote nodes
- Conclusion - terminating communication between the local and remote nodes

The following sections describe the contents of each of the parts of a user program that calls TOPS-10 X.25 subroutines.

2.4.1 Initiating an SVC Call and Using the SVC

When you write a program that initiates an SVC call, perform the following steps:

1. Initialization

   - Execute the INITIATE SWITCHED CIRCUIT function, which initiates an SVC call to a remote DTE and changes the port state to CALLING.
   - Execute the READ PORT STATUS function to determine if the port state has changed to CLEARED or RUNNING.
   - If the port state has become CLEARED, the remote DTE or remote node has rejected your call; execute the TERMINATE PORT ACCESS function before using the port again. If the port state has become RUNNING, the remote node has accepted your call, and your program can continue.

2. Sending and Receiving

   The TOPS-10 X.25 software includes a group of subroutines that:

   - Send and receive data
   - Send and read interrupts
   - Send and respond to reset requests

   Regardless of whether your program conducts communication over an SVC or a PVC, the same group of TOPS-10 X.25 subroutines is used to perform those functions. Table 2-2 lists this group of subroutines as the functions that you can use with SVCs and PVCs.

   When sending or receiving data:

   - Execute the SEND DATA MESSAGE function to send data
   - Execute the READ DATA MESSAGE function to receive data

   NOTE

   The TOPS-10 PSI Gateway maintains flow control of Data packets. This relieves you of the need to maintain the packet send (P(S)) and packet receive (P(R)) sequence numbers, which are discussed in Section 1.4.2.

   When sending or reading interrupt data:

   - Execute the SEND INTERRUPT MESSAGE function to send interrupt data. The X.25 software allows no more than one outstanding unacknowledged interrupt.
   - Execute the READ PORT STATUS function to determine if an interrupt has been acknowledged,
   - Execute the READ INTERRUPT MESSAGE function to obtain interrupt data transmitted by the remote node.

- Execute the CONFIRM INTERRUPT MESSAGE function to acknowledge receipt of an interrupt transmitted by a remote node.

Note that because flow control of interrupts occurs independent of data flow control, interrupts can overtake other transmitted data.

When handling resets:

- Execute the RESET VIRTUAL CIRCUIT function to request that a virtual circuit be reset or to acknowledge a reset.
- Use the READ PORT STATUS function to determine if a virtual circuit has been reset.
- Execute the READ RESET DATA function to obtain the cause and diagnostic codes from the Reset Indication Packet.

When a reset request has been made, data and interrupts that are in transit on the virtual circuit can be discarded.

3. Conclusion

- Execute the READ PORT STATUS function.
- If the port state is CLEARED, the remote node has cleared the SVC you have been using; execute the READ CLEAR DATA function to obtain any information the remote node sent when it cleared the SVC. If the port state is not CLEARED, execute the CLEAR SWITCHED CIRCUIT function to terminate the SVC you have been using.
- Execute the TERMINATE PORT ACCESS function, which releases the port you have been using.

2.4.2 Receiving an SVC Call and Using the SVC

When you write a program that is to receive an SVC call, perform the following steps:

1. Initialization

- Execute the WAIT INCOMING CALL function, which enables your job to receive a call.
- Execute the READ PORT STATUS function until the port state becomes CALLED.
- Execute the READ INCOMING CALL function, which obtains any data sent from the remote node when the call was made. Use that data to determine whether to accept or clear the call.
- Execute the ACCEPT INCOMING CALL or CLEAR SWITCHED CIRCUIT function.

2. Sending and Receiving

   - Send and receive data and interrupts. Send and respond to reset requests. For information on performing these functions, see item 2 in Section 2.4.1.

3. Conclusion

   - Execute the READ CLEAR DATA or CLEAR SWITCHED CIRCUIT function, then execute the TERMINATE PORT ACCESS function. For information on performing these functions, see item 3 in Section 2.4.1.

2.4.3 Using a PVC

When you write a program that uses a PVC, you must include the following steps:

1. Initialization

   - Execute the OPEN PERMANENT CIRCUIT function, which acquires the specified PVC for your exclusive use at your node.

   - Execute the READ PORT STATUS function to determine if the port state has changed to UNSYNC, RUNNING, or ERROR. If the port state has changed to UNSYNC, the PPSN or remote node has initiated a reset; execute the RESET VIRTUAL CIRCUIT function, and continue. If the port state is ERROR, the TOPS-10 PSI Gateway Software has rejected your open request; to recover, execute the TERMINATE PORT ACCESS function, then again execute the OPEN PERMANENT CIRCUIT function.

2. Sending and Receiving

   - Send and receive data and interrupts. Send and respond to reset requests. For information on performing these functions, see item 2 in Section 2.4.1.

3. Conclusion

   - Execute the TERMINATE PORT ACCESS, which releases both the PVC and the port you have been using.

CHAPTER 3

FORTRAN-10 SUBROUTINE CALLS

3.1 FORTRAN-10 AND THE TOPS-10 PSI GATEWAY ACCESS SOFTWARE

When issuing TOPS-10 PSI Gateway Access subroutine calls in FORTRAN-10 programs, you normally specify variable names for the subroutine arguments. You must also place values in some of the variables you have named before your FORTRAN-10 program is executed. In other cases, the subroutine returns values in variables you have named as the program is executed.

Discussions of the FORTRAN-10 TOPS-10 PSI Gateway Access subroutine calls in Section 3.2 explain which values you must supply and which values are returned by the software.

This section lists the types of variables you can specify in FORTRAN-10 TOPS-10 PSI Gateway Access subroutine calls.

These data types are not necessarily the standard FORTRAN-10 data types. For information on standard FORTRAN-10 data types, see the TOPS-10/20 FORTRAN Language Manual.

The data types are:

8-BIT BYTE	the rightmost 8 bits in a word, unless otherwise specified in Section 3.2.
8-BIT BYTE ARRAY	a 1-dimensional array composed of four 8-bit bytes per word. These 8-bit bytes are left justified in each word; the rightmost 4 bits in each word are unused.
INTEGER	a whole decimal number that occupies one word and is in the range (-2**35)-1 to (+2**35)-1.
INTEGER ARRAY	a 1-dimensional array composed of integers.
ASCII STRING	a string of contiguous 7-bit ASCII characters. These characters are left justified on the word boundary of the first word and terminated by a null character in the word following the last word of the string.
LOGICAL	a variable that can have a value of either .TRUE. or .FALSE.

The names of the subroutine arguments in Section 3.2 do not follow the FORTRAN-10 convention specifying that variable names beginning with the letters I, J, K, L, M, or N are integer variables. To determine whether a variable is an integer, read the description of that variable.

NOTE

Since FORTRAN-10 does not allow null parameters in an argument list, specify a null string with the value 0.

3.2 SENDING AND RECEIVING DATA

This section discusses how to send and receive data in a FORTRAN-10 program and how the FORTRAN-callable TOPS-10 PSI Gateway Access Routines convert data before and after transmission through a PPSN.

The subroutine that queues data for transmission to a PPSN is SEND DATA MESSAGE, and the subroutine that receives data transmitted through a PPSN is READ DATA MESSAGE. The FORTRAN-callable SEND DATA MESSAGE subroutine is X25SDM; the FORTRAN-callable READ DATA MESSAGE subroutine is X25RDM. The names of arguments used in calling X25SDM and X25RDM are noted throughout this section. See the discussions of X25SDM and X25RDM in Section 3.4 for more information about those arguments.

When calling X25SDM:

- Specify an array of data to be transmitted in the datbuf argument.
- Specify a value that indicates how the TOPS-10 PSI Gateway Access Routines should interpret the data to be transmitted in the dtype argument (See Table 3-1 for a list of values you can specify for dtype). The value you should specify for dtype depends on the destination for the data and the type of data being transmitted. To transmit 36-bit binary data to another TOPS-10 system, specify 0 for dtype (see Section 3.2.1). To transmit any type of data to another operating system (for example, one that has a word size other than 36 bits), specify a value other than 0 for dtype (see Sections 3.2.2 and 3.2.3).

When calling X25RDM:

- Specify an array to contain received data in the datbuf argument.
- Specify a value that indicates how received data is to be placed in datbuf in the dtype argument. When calling X25RDM, the value you should specify for dtype depends on the source of the data and the type of data being transmitted. For example, if 36-bit binary data was transmitted by a TOPS-10 system and dtype 0 was specified in the X25SDM call at that system, specify a value of 0 in the X25RDM call (see Section 3.2.1).

3.2.1 Conversion of 36-Bit Binary Data in TOPS-10/TOPS-10 Transfers

When you write programs to transmit 36-bit binary data between two TOPS-10 systems, specify a value of 0 for dtype in all X25SDM and X25RDM subroutine calls. Doing so ensures the integrity of data transmitted between the two TOPS-10 systems.

Calling X25SDM with dtype 0 causes the TOPS-10 PSI Gateway Access Routines to treat the entire array specified in datbuf as a stream of bits. Before transmitting this data to a PPSN, the Access Routines convert the data into eight-bit units called octets. This conversion is performed from left to right through the array specified in datbuf. The first bit in datbuf becomes the most significant (left-most) bit in the first octet. The eighth bit in datbuf becomes the least significant (right-most) bit in the first octet. The ninth bit becomes the most significant bit in the second octet, and so on. If the last octet formed contains fewer than eight bits, the Access Routines fill the remaining positions in that octet with binary zeros.

Calling X25RDM with dtype 0 causes the TOPS-10 PSI Gateway Access Routines to copy the data from received octets into successive bit positions in elements of the receiving datbuf array.

3.2.2 Other Types of Binary Data Conversion

When you write programs that send and/or receive non-36-bit binary data, specify dtype value 1, 2, 3, or 4. (See Table 3-1 for information about these dtype values.)

When you specify dtype value 1, 2, 3, or 4 in an X25SDM call, the TOPS-10 PSI Gateway Access Routines treat each element of the array specified in datbuf as a series of 8-bit bytes. The Access Routines format each 8-bit byte into an octet. This formatting occurs left to right through the datbuf array.

When you specify dtype value 1, 2, 3, or 4 in an X25RDM call, the Access Routines format the received octets into 8-bit bytes and place these 8-bit bytes in the datbuf array. This conversion occurs according to the value you give dtype. If the two programs transmitting and receiving data are both running on TOPS-10 systems, specify the same value for dtype in the X25RDM call as was used in the X25SDM call.

3.2.3 ASCII Data Conversion

When you write programs that send and/or receive 7-bit ASCII data, specify dtype value 5, 6, 7, 8, or 9. (See Table 3-1 for information about these dtype values.)

When you specify dtype value 5, 6, 7, 8, or 9 in an X25SDM call, the TOPS-10 PSI Gateway Access Routines format each 7-bit ASCII character from the datbuf array into one octet. The Access Routines set the most significant (left-most) bit in the octet to 0. This formatting occurs left to right through the datbuf array.

When you specify dtype value 5, 6, 7, 8, or 9 in an X25RDM call, the Access Routines format the received octets into 7-bit bytes in the datbuf array. The number of 7-bit bytes per array element depends upon the value you specify for dtype. The Access Routines ignore the most significant (left-most) bit of each received octet.

3.3 LINKING A FORTRAN-10 PROGRAM

The FORTRAN-10 TOPS-10 PSI Gateway Access Routines are in the library file X25GAF.REL. After you have written a program and compiled it to produce relocatable object modules, you must link the modules with X25GAF.REL to produce an executable image file. To link your program, type:

```
.R LINK (RET)
*REL:X25GAF.REL, object-1[, ..., object-n] (RET)
*/SAVE /GO (RET)
```

where:

object-1 through object-n are the names of your relocatable object modules.

3.4 THE SUBROUTINE CALLS

The TOPS-10 PSI Gateway Access Routine functions summarized in Chapter 2 are described here as a set of FORTRAN-10 subroutine calls. The calls are in alphabetical order in the following format:

- A title that contains the name of the FORTRAN-10 call and the TOPS-10 PSI Gateway Access Routine function
- Information on how to use the function
- The call format
- Descriptions of the subroutine arguments

X25AIC ACCEPT INCOMING CALL

Use the X25AIC subroutine to accept an incoming call request. You can use data returned by the X25RIC (READ INCOMING CALL) subroutine to determine whether to accept an incoming call. Before you issue an X25AIC subroutine call, the port state must be CALLED.

The format of the X25AIC subroutine call is:

```
CALL X25AIC (nport, usrgrp, facbuf, faclen, datbuf, datlen,
             rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you call X25WIC (WAIT INCOMING CALL) to set up the virtual circuit.

usrgrp — is a user-supplied ASCII string that contains the name of a bilateral closed user group or closed user group. This name can be a maximum of 16 ASCII characters in length. If you choose not to supply the user group, specify 0 as the usrgrp argument.

facbuf — is an 8-bit-byte array in which you place any facilities data needed to support your PPSN service. Specify the length of facbuf in faclen. (For information on facilities data, see the documentation for your PPSN.)

faclen — is a user-supplied integer that specifies the length of facbuf in 8-bit bytes. The maximum value for faclen depends upon the value you specify in usrgrp for this subroutine call:

- If you specify a bilateral closed user group in usrgrp, the maximum value for faclen is 60.
- If you specify a closed user group in usrgrp, the maximum value for faclen is 61.
- If you do not specify a value for user group, the maximum value for faclen is 63.

Your PPSN may further restrict this field.

datbuf — is an 8-bit-byte array in which you place user-specified accept data. Specify the length of datbuf in datlen.

datlen — is a user-supplied integer that specifies the length of datbuf in 8-bit bytes. The maximum value for datlen normally is 16 (or 128 for fast select calls), although your PPSN may further restrict this field.

X25AIC (Cont.)

rcode — is an integer variable in which X25AIC stores a return code. If rcode is nonzero, the incoming call is not accepted. Rcode values and their meanings are:

0 The X25AIC subroutine is executed successfully: the incoming call is accepted, and the port state changes from CALLED to RUNNING.

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25WIC before executing X25CIM.
- You have attempted to execute this subroutine when the port state is not CALLED.
- The logical link between the user job and the Gateway node has been disconnected.

7 The facilities data field is truncated because it is longer than the limit set by the PPSN.

8 The user-specified accept data field is truncated because it is longer than the limit set by the PPSN.

X25CIM CONFIRM INTERRUPT MESSAGE

Use the X25CIM subroutine to confirm the receipt of interrupt data. When you issue an X25CIM subroutine call, the port state must be RUNNING. Only one interrupt can be outstanding at a time.

The format of the X25CIM subroutine call is:

```
CALL X25CIM (nport, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you set up the virtual circuit by calling X25ISC (INITIATE SWITCHED CIRCUIT), X25OPC (OPEN PERMANENT CIRCUIT), or X25WIC (WAIT INCOMING CALL).

rcode — is an integer variable in which X25CIM stores a return code. If rcode is nonzero, the interrupt is not confirmed. Rcode values and their meanings are:

0 The X25CIM subroutine is executed successfully: the received interrupt is confirmed. The port state remains RUNNING.

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25ISC, X25OPC, or X25WIC before executing X25CIM.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- The logical link between the user job and the Gateway node has been disconnected.

12 No interrupt confirmation is requested. The PPSN does not send you an interrupt message that requires confirmation.

X25CSC CLEAR SWITCHED CIRCUIT

Use the X25CSC subroutine to terminate an SVC connection. Once you have executed the X25CSC subroutine, any data in transit can be lost, and you can no longer communicate over the specified circuit. However, the port remains assigned; to release the port, execute the X25TPA (TERMINATE PORT ACCESS) subroutine. When you issue an X25CSC subroutine call, the port state must be CALLED, CALLING, RUNNING, SYNC, or UNSYNC.

The format of the X25CSC subroutine call is:

```
CALL X25CSC (nport, usrclr, usrgrp, facbuf, faclen, datbuf,
             datlen, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you call X25ISC (INITIATE SWITCHED CIRCUIT) or X25WIC (WAIT INCOMING CALL) to set up the virtual circuit.

usrclr — is a user-supplied 8-bit byte in which you place the clear diagnostic for transmission to the remote DTE. Users of the SVC choose values for the clear diagnostic, which is the reason for clearing the SVC.

usrgrp — is a user-supplied ASCII string that contains the name of a bilateral closed user group or closed user group. This name can be a maximum of 16 ASCII characters in length.

facbuf — is an 8-bit-byte array in which you place any facilities data required to support your PPSN service. Specify the length of facbuf in faclen. (For information on facilities data, see the documentation for your PPSN.)

faclen — is a user-supplied integer that specifies the length of facbuf in 8-bit bytes. The maximum value for faclen depends upon the value you have specified in usrgrp for this subroutine call:

- If you have specified a bilateral closed user group in usrgrp, the maximum value for faclen is 60.
- If you have specified a closed user group in usrgrp, the maximum value for faclen is 61.
- If you have not specified a value for user group, the maximum value for faclen is 63.

Your PPSN may further restrict this field.

datbuf — is an 8-bit-byte array in which you place user-specified clear data. Specify the length of datbuf in datlen.

X25CSC (Cont.)

datlen — is a user-supplied integer that specifies the length of datbuf in 8-bit bytes. The maximum value for datlen normally is 16 (or 128 for fast select calls), although your PPSN may further restrict this field.

rcode — is an integer variable in which X25CSC stores a return code. If rcode is nonzero, the SVC is not cleared. Rcode values and their meanings are:

0 The X25CSC subroutine is executed successfully: a clear request is sent to the PPSN, and the port state changes from RUNNING to CLEARING.

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25ISC or X25WIC before executing X25CSC.
- You have attempted to execute this subroutine when the port state is not CALLING, CALLED, RUNNING, SYNC, or UNSYNC.
- The logical link between the user job and the Gateway node has been disconnected.

7 The facilities data field is truncated because it is longer than the limit set by the PPSN.

8 The user-specified clear data field is truncated because it is longer than the limit set by the PPSN.

X25ISC INITIATE SWITCHED CIRCUIT

Use the X25ISC subroutine to issue a virtual call request over a switched virtual circuit to a remote DTE. A successful return does not indicate that the remote DTE has responded to the call. When the remote DTE responds to the call, the port state changes from CALLING to RUNNING (if the remote user accepted the call) or CLEARED (if the remote user rejected the call). When you issue an X25ISC subroutine call, the port state must be UNDEFINED.

The format of the X25ISC subroutine call is:

```
CALL X25ISC (netwrk, passwd, dest, subadr, usrgrp, facbuf,
             faclen, datbuf, datlen, buffer, nport, rcode)
```

where:

netwrk — is a user-supplied ASCII string that contains the name of the PPSN over which you want to communicate. This name can be a maximum of 16 ASCII characters.

passwd — is a user-supplied ASCII string that contains the password required for gaining access to the X.25 Gateway. This password can be a maximum of 16 ASCII characters. Your system manager determines the X.25 Gateway Access password. If you do not need to use a password, specify 0 for the passwd argument.

dest — is a user-supplied ASCII string that contains a numeric string representing the address of the called DTE. This address can be a maximum of 15 ASCII characters.

subadr — is a user-supplied ASCII string that contains a numeric string representing the address of the calling DTE. This address can be a maximum of 15 ASCII characters. If you choose not to supply the source-DTE address, specify 0 for the subadr argument.

usrgrp — is a user-supplied ASCII string that contains the name of a bilateral closed user group or closed user group. This name can be a maximum of 16 ASCII characters. If you choose not to supply the user group, specify 0 as the usrgrp argument.

facbuf — is an 8-bit-byte array in which you place any facilities data required to support your PPSN service. Specify the length of facbuf in faclen. (For information on facilities data, see the documentation for your PPSN.)

faclen — is a user-supplied integer that specifies the length of facbuf in 8-bit bytes. The maximum value for faclen depends upon the value you have specified in usrgrp for this subroutine call:

- If you have specified a bilateral closed user group in usrgrp, the maximum value for faclen is 60.

X25ISC (Cont.)

- If you have specified a closed user group in usrgrp, the maximum value for faclen is 61.
- If you have not specified a value for user group, the maximum value for faclen is 63.

Your PPSN may further restrict this field.

datbuf — is an 8-bit-byte array in which you place user-specified call data. Specify the length of datbuf in datlen.

datlen — is a user-supplied integer that specifies the length of datbuf in 8-bit bytes. The maximum value for datlen normally is 16 (or 128 for fast select calls).

buffer — is an integer array that is used as workspace by the X.25 Gateway Access Routines. This buffer can be 144 to 396 words, depending on your selected packet size. To compute the length of the buffer, use the following formula:

```
length = 140 + (packet size/4)
```

For example, if your packet size is 16, specify a buffer length of 144.

nport — is an integer variable in which X25ISC returns the assigned port identifier. You must use this value as an argument for many of the other subroutine calls.

rcode — is an integer variable in which X25ISC stores a return code. If rcode is nonzero, the switched circuit is not initiated, and no port is assigned. Values for rcode and their meanings are:

0 X25ISC is executed successfully; a call request is sent to the PPSN. The port state changes from UNDEFINED to CALLING.

1 THE TOPS-10 PSI Gateway does not have sufficient resources to allocate a new switched circuit.

2 The TOPS-10 PSI Gateway Access Routines do not have sufficient resources to allocate a new circuit.

X25ISC (Cont.)

3 The TOPS-10 PSI Gateway software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has already been allocated.
- A logical link could not be established to the Gateway node for reasons other than those specified here.

6 You have specified a network name that has not been defined in the DECnet-10 Network Management data base.

7 The facilities data field is truncated because it is longer than the limit set by the PPSN.

8 The user-specified call data field is truncated because it is longer than the limit set by the PPSN.

9 You did not specify a password, or you specified an incorrect password for gaining access to the PPSN through the Gateway node.

10 You did not specify the destination DTE address to which the circuit is to be established.

13 The specified destination DTE address is too long.

14 The specified source DTE address is too long.

17 The version numbers of the TOPS-10 PSI Gateway Access Routines and TOPS-10 PSI Gateway Software are not compatible.

X25NCS NO COMMUNICATION SEEN

Use the X25NCS subroutine to acknowledge one or more failures of a permanent virtual circuit due to loss of communication with the PPSN. After such a failure, you retain possession of the permanent virtual circuit. When you issue an X25NCS call, the port state must be NO COMMUNICATION.

The format of the X25NCS subroutine call is:

```
CALL X25NCS (nport, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you call the subroutine X25OPC (OPEN PERMANENT CIRCUIT) to set up the virtual circuit.

rcode — is an integer variable in which X25NCS stores a return code. If rcode is nonzero, the subroutine fails to acknowledge the loss of communication with the PPSN. Rcode values and their meanings are:

0 The X25NCS subroutine is executed successfully: you have acknowledged a loss of communication with the PPSN and are ready to resume using the permanent virtual circuit, and the port state changes from NO COMMUNICATION to RUNNING.

3 The TOPS-10 PSI Gateway Access Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25OPC before executing X25NCS.
- You have attempted to execute this subroutine when the port state is not NO COMMUNICATION.
- You have attempted to execute this subroutine with a switched virtual circuit.
- The logical link between the user job and the Gateway node has been disconnected.

X25OPC OPEN PERMANENT CIRCUIT

Use the X25OPC subroutine to acquire the exclusive use of a specified PVC. The function does not try to contact the remote DTE. A successful response changes the port state to RUNNING or UNSYNC. You can transfer data over the PVC as soon as the port state is RUNNING.

A port state of UNSYNC indicates that the remote DTE has issued a reset; you must issue an X25RVC (RESET VIRTUAL CIRCUIT) subroutine call before you can proceed. When you issue an X25OPC subroutine call, the port state must be UNDEFINED.

The format of the X25OPC subroutine call is:

```
CALL X25OPC (netwrk, passwd, pvcnam, buffer, nport, rcode)
```

where:

netwrk — is a user-supplied ASCII string that contains the name of the PPSN over which you wish to communicate. This name can be a maximum of 16 ASCII characters.

passwd — is a user-supplied ASCII string that contains the required password for gaining access to the X.25 Gateway. This password can be a maximum of 16 ASCII characters. Your system manager determines the X.25 Gateway Access password. If you do not need to use a password, specify a passwd of 0.

pvcnam — is a user-supplied ASCII string that contains the name of the PVC you are using. This name can be a maximum of 16 ASCII characters. Your system manager can supply you with the PVC name.

buffer — is an integer array that is used as workspace by the X.25 Gateway Access Routines. This array can range in size from 144 to 396 words, depending on your packet size. To compute the length of the buffer, use the following formula:

```
length = 140 + (packet size/4)
```

For example,if your packet size is 16, specify a buffer length of 144.

nport — is an integer variable in which X25OPC returns the assigned port identifier. You must use this value as an argument for many of the other subroutine calls.

X25OPC (Cont.)

rcode — is an integer variable in which X25OPC stores a return code. If rcode is nonzero, you have not obtained access to the specified PVC. Rcode values and their meanings are:

0 X25OPC is executed successfully: a permanent virtual circuit was allocated exclusively for your job, and the port state changes from UNDEFINED to OPEN.

1 The TOPS-10 PSI Gateway does not have sufficient resources to allocate a new permanent circuit.

2 The TOPS-10 PSI Gateway Access Routines do not have sufficient resources to allocate a new circuit.

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has already been allocated.
- A logical link could not be established to the Gateway node for reasons other than those specified here.

6 You have specified a network name that has not been defined in the DECnet-10 Network Management data base.

9 You have not specified a password, or you have specified an incorrect password for gaining access to the PPSN through the Gateway node.

10 You have specified a permanent circuit that has not been defined in the DECnet-10 Network Management data base.

17 The version numbers of the TOPS-10 PSI Gateway Access Routines and TOPS-10 PSI Gateway Software are not compatible.

X25RAD READ ACCEPT DATA

Use the X25RAD subroutine to read information obtained from the remote DTE after it accepted your request to establish a switched virtual circuit connection. Some PPSNs do not supply data in all the described fields. Where no data is supplied, the fields are filled with nulls (binary 0s) or the corresponding length field is zero. When you issue an X25RAD subroutine call, the port state must be RUNNING.

The format of the X25RAD subroutine call is:

```
CALL X25RAD (nport, usrgrp, facbuf, faclen, datbuf, datlen,
             rcode)
```

where:

nport is an integer that identifies the port you are using. A value for nport is returned when you call X25ISC (INITIATE SWITCHED CIRCUIT) to set up the virtual circuit.

usrgrp is an ASCII string in which X25RAD returns the binary closed user group or closed user group supplied by the remote DTE. This field may be a maximum of 16 ASCII characters and is delimited by a binary 0.

facbuf is an 8-bit-byte array in which X25RAD returns facilities data supplied by the remote DTE. The length of facbuf is specified in faclen. (For information on facilities data, see the documentation for your PPSN.)

faclen is an integer variable that you set to the maximum number of 8-bit bytes facbuf can receive. This data can be as many as 63 8-bit bytes. Upon return, X25RAD sets faclen equal to the actual number of 8-bit bytes returned in facbuf.

datbuf is an 8-bit-byte array in which X25RAD returns accept data supplied by the remote DTE. The length of datbuf is specified in datlen.

datlen is an integer variable that you set to the maximum number of 8-bit bytes datbuf can accept. This data can be as many as 16 8-bit bytes (or 128 for fast select calls). Upon return, X25RAD sets faclen to the actual number of 8-bit bytes returned in datbuf.

rcode is an integer variable in which X25RAD stores a return code. If rcode is nonzero, the subroutine has failed to retrieve all data supplied by the remote DTE. Rcode values and their meanings are:

0 The X25RAD subroutine is executed successfully: the user accept data and/or user accept facilities were returned. The port state remains RUNNING.

X25RAD (Cont.)

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25ISC before executing X25RAD.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- You have attempted to execute this subroutine with a permanent virtual circuit.
- The logical link between the user job and the Gateway node has been disconnected.

4 There is no user-specified accept data.

7 The facilities data field was truncated because you did not supply a buffer large enough to accept that data.

8 The user-specified accept data field was truncated because you did not supply a buffer large enough to accept that data.

X25RCD READ CLEAR DATA

Use the X25RCD subroutine to read data obtained when an SVC is cleared. The content of the data is PPSN specific, but not all PPSNs provide this information when an SVC is cleared. Use the X25RPS (READ PORT STATUS) subroutine to determine that the PPSN has cleared the virtual circuit. When you issue an X25RCD subroutine call, the port state must be CLEARED.

The format of the X25RCD subroutine call is:

```
CALL X25RCD (nport, cause, diag, usrgrp, facbuf, faclen, datbuf,
             datlen, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you call X25ISC (INITIATE SWITCHED CIRCUIT) or X25WIC (WAIT INCOMING CALL) to set up the virtual circuit.

cause — is an integer variable in which X25RCD returns the network clear cause supplied by the PPSN. If cause is nonzero, the network has cleared the call, cause contains the network clear cause, and diag contains the network clear diagnostic. If cause is 0, the remote user has cleared the SVC, and the user clear diagnostic is contained in diag. (For information on the network clear cause and diagnostic, see the documentation for your PPSN.)

diag — is an integer variable in which X25RCD returns the network or user clear diagnostic.

usrgrp — is an ASCII string into which X25RCD returns the binary closed user group or closed user group supplied by the remote DTE. This field may be a maximum of 16 ASCII characters and is delimited by a binary 0.

facbuf — is an 8-bit-byte array in which X25RCD returns facilities data supplied by the remote DTE. The length of facbuf is specified in faclen. (For information on facilities data, see the documentation for your PPSN.)

faclen — is an integer variable that you set to the maximum number of 8-bit bytes facbuf can receive. This data can be as many as 63 8-bit bytes. Upon return, X25RAD sets faclen equal to the actual number of 8-bit bytes returned in facbuf.

datbuf — is an 8-bit-byte array in which X25RCD returns clear data supplied by the remote DTE. The length of datbuf is specified in datlen.

X25RCD (Cont.)

datlen — is an integer variable that you set to the maximum number of 8-bit bytes datbuf can accept. This data can be as many as 16 8-bit bytes (or 128 for fast select calls). Upon return X25RCD sets faclen to the actual number of 8-bit bytes returned in datbuf.

rcode — is an integer variable in which X25RCD stores a return code. If rcode is nonzero, the subroutine has failed to retrieve data supplied by the remote DTE. Rcode values and their meanings are:

0 The X25RCD subroutine is executed successfully: the clear cause, clear diagnostic, and/or user clear data and/or facilities data were returned. The port state remains CLEARED.

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25ISC or X25WIC before executing X25RCD.
- You have attempted to execute this subroutine when the port state is not CLEARED.
- You have attempted to execute this subroutine with a permanent virtual circuit.
- The logical link between the user job and the Gateway node has been disconnected.

7 The facilities data field was truncated because you did not supply a buffer large enough to contain this data.

8 The clear data field was truncated because you did not supply a buffer large enough to contain this data.

X25RDM READ DATA MESSAGE

Use the X25RDM subroutine to receive data from a virtual circuit. The data can be either normal or qualified. When you issue an X25RDM subroutine call, the port state must be RUNNING.

The format of the X25RDM subroutine call is:

```
CALL X25RDM (nport, dtype, datbuf, datlen, qbit, mbit, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you set up the virtual circuit by calling X25ISC (INITIATE SWITCHED CIRCUIT), X25OPC (OPEN PERMANENT CIRCUIT), or X25WIC (WAIT INCOMING CALL).

dtype — is an integer that you set to a value indicating how received data should be placed in datbuf. See Table 3-1 for a list of values you can specify for dtype.

datbuf — is an array of any data type in which X25RDM returns received data. X25RDM places data in the array according to the value of dtype.

datlen — is an integer that you set to the maximum length of datbuf. To avoid a truncation error (rcode error 8), use the minimum value for datlen that corresponds to the value you have specified for dtype (see Table 3-1). You can obtain your packet size (ps) by executing the X25RPS (READ PORT STATUS) subroutine. Upon return, X25RDM sets datlen to the actual length of datbuf in the units specified in dtype. If the returned actual length in datlen is 0 and the return code in rcode is 0 (successful), you have received an empty Data packet.

qbit — is an integer in which X25RDM returns a value indicating data qualification. If qbit is 1, datbuf contains qualified data.

mbit — is a logical variable in which X25RDM returns a value indicating whether the next packet to be received is logically related to the current one. If mbit is .TRUE., the next packet is logically related to the current one (the nature of this relationship is user defined). If mbit is .FALSE., the next packet is not logically related to the current one. (See the discussion of X25SDM for information on the relationship between mbit and the more bit of the received Data packet.)

X25RDM (Cont.)

Table 3-1: Specifications for the Dtype and Datlen Variables

Dtype Values	Datlen Minimum Values	Unit Size	Location of data in Each Array Element
0	(ps*2/9)+1 †	36	All 36 bits
1	ps/4	32	Leftmost 32 bits
2	ps/4	32	Rightmost 32 bits
3	ps/2	16	Rightmost 16 bits
4	ps	8	Rightmost 8 bits
5	ps	7	Leftmost 7 bits, which is equivalent to the A1 FORMAT field descriptor - 1 unit per array element
6	ps/2	7	Leftmost 14 bits, which is equivalent to the A2 FORMAT field descriptor - 2 units per array element
7	(ps/3)+1	7	Leftmost 21 bits, which is equivalent to the A3 FORMAT field descriptor - 3 units per array element
8	ps/4	7	Leftmost 28 bits, which is equivalent to the A4 FORMAT field descriptor - 4 units per array element
9	(ps/5)+1	7	Leftmost 35 bits, which is equivalent to the A5 FORMAT field descriptor - 5 units per array element

† ps is your packet size

rcode is an integer variable in which X25RDM stores a return code. If rcode is nonzero, the subroutine has failed to receive data from the virtual circuit. Rcode values and their meanings are:

0 The X25RDM subroutine is executed successfully: a normal or qualified user data packet was returned. The port state remains RUNNING.

X25RDM (Cont.)

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25ISC, X25OPC, or X25WIC before executing X25RDM.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- The logical link between the user job and the Gateway node has been disconnected.

4 There is no user data to be returned.

8 The received data was truncated because you have supplied a buffer too small to contain an entire data packet.

16 You have received a partially filled data packet with the more bit set. The more bit is set upon return from the function. However, it is entirely up to you how you handle such a data packet. For example, you may choose to ignore the data packet and reset the circuit.

18 There is no data from the PPSN to be returned because the circuit has been reset.

19 You have specified an invalid value for dtype.

X25RIC READ INCOMING CALL

Use the X25RIC subroutine to obtain information about an incoming call when the port state changes from LISTENING to CALLED. When you issue an X25RIC subroutine call, the port state must be CALLED.

The format of the X25RIC subroutine call is:

```
CALL X25RIC (nport, netwrk, addr, subadr, usrgrp, facbuf,
             faclen, datbuf, datlen, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you call X25WIC (WAIT INCOMING CALL) to set up the virtual circuit.

netwrk — is an ASCII string in which the X25RIC subroutine places the name of the network from which the call originated. This name can be a maximum of 16 ASCII characters and is delimited by a binary 0.

addr — is an ASCII string in which the X25RIC subroutine places the address of the calling (remote) DTE. This address can be a maximum of 16 ASCII characters and is delimited by a binary 0.

subadr — is an ASCII string in which this subroutine places the address of the called (local) DTE. This address can be a maximum of 16 ASCII characters and is delimited by a binary 0.

usrgrp — is an ASCII string in which this subroutine places the binary closed user group or closed user group supplied by the remote DTE. This field may be a maximum of 16 ASCII characters and is delimited by a binary 0.

facbuf — is an 8-bit-byte array in which X25RIC returns facilities data supplied by the remote DTE. The length of facbuf is specified in faclen. (For information on facilities data, see the documentation for your PPSN.)

faclen — is an integer variable that you set to the maximum number of 8-bit bytes facbuf can receive. This data can be as many as 63 8-bit bytes. Upon return, X25RIC sets faclen equal to the actual number of 8-bit bytes returned in facbuf.

datbuf — is an 8-bit-byte array in which this subroutine returns user-specified call data supplied by the remote DTE. The length of datbuf is specified in datlen.

X25RIC (Cont.)

datlen — is an integer variable that you set to the maximum number of 8-bit bytes datbuf can accept. This data can be as many as 16 8-bit bytes (or 128 for fast select calls). Upon return, X25RIC sets faclen to the actual number of 8-bit bytes returned in datbuf.

rcode — is an integer variable in which X25RIC stores a return code. If rcode is nonzero, the subroutine has failed to obtain information about an incoming call. Rcode values and their meanings are:

0 The X25RIC subroutine is executed successfully: incoming call data and/or facilities data were returned. The port state remains CALLED.

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25WIC before executing X25RIC.
- You have attempted to execute this subroutine when the port state is not CALLED.
- The logical link between the user job and the Gateway node has been disconnected.

4 There is no user-specified call data.

7 The facilities data field was truncated because you did not supply a buffer large enough to contain that data.

8 The user-specified call data field was truncated because you did not supply a buffer large enough to contain that data.

X25RIM READ INTERRUPT MESSAGE

Use the X25RIM subroutine to receive interrupt data. When you issue an X25RIM subroutine call, the port state must be RUNNING. Use the X25RPS (READ PORT STATUS) subroutine to determine if there is an interrupt byte to read.

The format of the X25RIM subroutine call is:

```
CALL X25RIM (nport, intbyt, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you set up the virtual circuit by calling X25ISC (INITIATE SWITCHED CIRCUIT), X25OPC (OPEN PERMANENT CIRCUIT), or X25WIC (WAIT INCOMING CALL).

intbyt — is an integer variable in the rightmost 8 bits of which X25RIM returns the interrupt byte.

rcode — is an integer variable in which X25RIM stores a return code. If rcode is nonzero, the subroutine has failed to obtain interrupt data. Values for rcode and their meanings are:

0 The X25RIM subroutine is executed successfully: an interrupt data byte was returned. The port state remains RUNNING.

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25ISC, X25OPC, or X25WIC before executing X25RIM.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- The logical link between the user job and the Gateway node has been disconnected.

5 There is no interrupt.

18 There is no interrupt from the PPSN to be returned because the circuit has been reset.

X25RPS READ PORT STATUS

Use the X25RPS subroutine to poll the port maintained by the TOPS-10 PSI Gateway Software. You can issue an X25RPS subroutine call regardless of the current port state. However, if the port state is UNDEFINED, perror, iiostd, oiostd, davail, iavail, and psize have indeterminate meaning.

The format of the X25RPS subroutine call is:

```
CALL X25RPS (nport, perror, stat, iiostd, oiostd, davail,
             iavail, psize, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you set up the virtual circuit by calling X25ISC (INITIATE SWITCHED CIRCUIT), X25OPC (OPEN PERMANENT CIRCUIT), or X25WIC (WAIT INCOMING CALL).

perror — is an integer variable in which X25RPS returns the last fatal error condition (see Table 3-2) that changed the port state to the ERROR state. Perror has meaning only when stat is 10 (ERROR).

stat — is an integer variable in which X25RPS returns the current port state as a decimal value. See Table 2-3 for a list of the decimal values and explanations of port states.

iiostd — is a logical variable in which X25RPS returns the incoming interrupt flag. If iiostd is .TRUE., there is a previously received interrupt that you have not yet confirmed; use X25CIM (CONFIRM INTERRUPT MESSAGE) to confirm the interrupt.

oiostd — is a logical variable in which X25RPS returns the outgoing interrupt flag. If oiostd is .TRUE., an outgoing interrupt has been transmitted but not confirmed by the remote DTE.

davail — is a logical variable in which X25RPS returns the data-available flag. If davail is .TRUE., incoming user data is available on the data subchannel; use X25RDM (READ DATA MESSAGE) to read the data.

X25RPS (Cont.)

iavail — is logical variable in which X25RPS returns the interrupt-data-available flag. If iavail is .TRUE., incoming interrupt data is available on the interrupt subchannel; use X25RIM (READ INTERRUPT MESSAGE) to read the data.

psize — is an integer variable in which X25RPS returns the current packet size.

rcode — is an integer variable in which X25RPS stores a return code. If rcode is nonzero, X25RPS has failed to poll the specified port. Rcode values and their meanings are:

- 0 — The X25RPS subroutine is executed successfully: the port status and flags have been returned. The port state remains unchanged.
- 3 — The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs because the logical link between the user job and the Gateway node has been disconnected.

X25RPS (Cont.)

Table 3-2: Fatal Error Conditions

Decimal Value	Meaning
1	Unknown error
2	Insufficient gateway resources
3	Circuit in use (reserved)
4	Undefined circuit name
5	Undefined network name
6	No communication with the Public Network
7	Data field truncated
8	Facilities field truncated
9	Source DTE subaddress is too long
10	Destination DTE address is too long
11	Version skew
12	Invalid user group

X25RRD READ RESET DATA

Use the X25RRD subroutine to read information obtained when a virtual circuit is reset. The content of the data is network specific, but not all networks provide this information when a call is reset. Use the X25RPS (READ PORT STATUS) subroutine to determine that the virtual circuit has been reset. When you issue an X25RRD subroutine call, the port state must be UNSYNC.

The format of the X25RRD subroutine call is:

```
CALL X25RRD (nport, cause, diag, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you set up the virtual circuit by calling X25ISC (INITIATE SWITCHED CIRCUIT), X25OPC (OPEN PERMANENT CIRCUIT), or X25WIC (WAIT INCOMING CALL).

cause — is an integer variable in which X25RRD returns the network reset cause supplied by the PPSN. If cause is nonzero, the network has reset the virtual circuit, cause contains the network reset cause, and diag contains the network reset diagnostic. If cause is 0, the remote user has reset the virtual circuit, and the user reset diagnostic is contained in diag. (For information on the network reset cause and diagnostic, see the documentation for your PPSN.)

diag — is an integer in which X25RRD returns the user reset diagnostic if the remote user has reset the virtual circuit.

rcode — is an integer variable in which X25RRD stores a return code. If rcode is nonzero, the subroutine has failed to retrieve data associated with the resetting of a virtual circuit. Rcode values and their meanings are:

0 The X25RRD subroutine is executed successfully: reset cause and reset diagnostic bytes were returned. The port state remains UNSYNC.

X25RRD (Cont.)

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25ISC, X25OPC, or X25WIC before executing X25RRD.
- You have attempted to execute this subroutine when the port state is not UNSYNC.
- The logical link between the user job and the Gateway node has been disconnected.

X25RVC RESET VIRTUAL CIRCUIT

Use the X25RVC subroutine to reinitialize a virtual circuit and return it to the state it was in before data was transferred over that circuit. At the time of resetting, the PPSN might discard data and interrupt packets that are in transit. Also, use the X25RVC subroutine to acknowledge a reset from the network.

When you issue an X25RVC subroutine call, the port state must be either RUNNING or UNSYNC. When you initiate a reset, the port state changes from RUNNING to SYNC. When the PPSN initiates a reset, the port state changes from RUNNING to UNSYNC. Acknowledge a PPSN reset as soon as possible because the network may time out and clear the circuit before you acknowledge. Acknowledging a PPSN reset changes the port state from UNSYNC to RUNNING.

The format of the X25RVC subroutine call is:

```
CALL X25RVC (nport, diag, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you set up the virtual circuit by calling X25ISC (INITIATE SWITCHED CIRCUIT), X25OPC (OPEN PERMANENT CIRCUIT), or X25WIC (WAIT INCOMING CALL).

diag — is an integer that you fill with one byte of diagnostic data. You and the remote user should agree upon acceptable values for diag. The X.25 software ignores this value if you use X25RVC to confirm a network reset.

rcode — is an integer variable in which X25RVC stores a return code. Rcode values and their meanings are:

0 X25RVC is executed successfully. If you used X25RVC to initiate a reset on the circuit, a reset request was sent to the PPSN and the port state changes from RUNNING to SYNC. If you used X25RVC to confirm a reset indication from the PPSN, a reset confirmation was sent to the PPSN, and the port state changes from UNSYNC to RUNNING.

X25RVC (Cont.)

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25ISC, X25OPC, or X25WIC before executing X25RVC.
- You have attempted to execute this subroutine when the port state is not RUNNING or UNSYNC.
- The logical link between the user job and the Gateway node has been disconnected.

X25SDM SEND DATA MESSAGE

Use the X25SDM subroutine to queue a buffer of data for transmission to the PPSN. When calling X25SDM, specify in the dtype variable how data in the array datbuf is to be interpreted.

If this subroutine is executed successfully, the X.25 Gateway Access Routines have successfully transmitted the buffer. However, successful execution does not mean that the data has reached the PPSN.

With X25SDM you can send data over either a normal or a qualified data subchannel.

When you call X25SDM, the TOPS-10 PSI Gateway Access Routines format the datbuf array into octets, the contents of which are determined by the value you specify for dtype (see Sections 3.2.1 through 3.2.3). Before transmitting the octets to a PPSN, the Access Routines form the octets into Data packets. During that conversion, if a Data packet becomes full, the Access Routines set the more bit of that packet - regardless of the value you have specified in the mbit parameter. The Access Routines then transmit the Data packet to the PPSN. This conversion continues in that way until all data in the datbuf array has been converted into Data packets and transmitted.

However, the value you specify for mbit does affect whether the Access Routines set the more bit of the final packet in the sequence of packets currently being transmitted. If you have specified a value of .FALSE. for mbit, the Access Routines transmit the final packet with the more bit not set. If you have specified a value of .TRUE. for mbit, and the final packet is full, the Access Routines transmit the final packet with the more bit set. If you have specified a value of .TRUE. for mbit, but the final packet is not full, the Access Routines do not immediately transmit the final packet. Such a packet normally is transmitted to the PPSN only after subsequent X25SDM calls provide additional data to be transmitted.

You can, however, cause the Access Routines to transmit a partially filled Data packet to the PPSN. To do so, call X25SDM, with mbit specified as .FALSE., and datlen set to 0. Note that if you call X25SDM in this way and there is no partially filled Data packet waiting to be sent, nothing is transmitted, but the subroutine is successfully executed (rcode is 0). Also, calling X25SDM with datlen specified as 0 and mbit specified as .TRUE. is meaningless and results in no packet being transmitted to the PPSN.

When you issue an X25SDM subroutine call, the port state must be RUNNING.

The format of the X25SDM subroutine call is:

```
CALL X25SDM (nport, dtype, datbuf, datlen, qbit, mbit, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you set up the virtual circuit by calling X25ISC (INITIATE SWITCHED CIRCUIT), X25OPC (OPEN PERMANENT CIRCUIT) or X25WIC (WAIT INCOMING CALL).

X25SDM (Cont.)

dtype — is an integer that you set to a value indicating how data in the transmitted array should be interpreted. See Table 3-1 for a list of dtype values.

datbuf — is an array of any data type in which you place data to be transmitted. This data is interpreted in the format specified in dtype.

datlen — is an integer that you have set to the length of datbuf. If you have specified a value for dtype in the range 5 through 9, then specify datlen as the number of characters in datbuf. Otherwise, specify datlen as the length of the datbuf array.

qbit — is an integer that you have set to indicate data qualification. If qbit is 0, datbuf contains normal data. If qbit is 1, datbuf contains qualified data.

mbit — is a logical variable that indicates whether the next datbuf you send is logically related to the current one. If mbit is .TRUE., the next datbuf is logically related to the current one (the nature of this relationship is user defined). If mbit is .FALSE., the next datbuf is not logically related to the current one. (See the introductory information on X25SDM for a discussion of the relationship between mbit and the more bit of the transmitted Data packet.)

rcode — is an integer variable in which X25SDM stores a return code. If rcode is nonzero, the subroutine has failed to transmit data. Rcode values and their meanings are:

0 The X25SDM subroutine is executed successfully: the data in datbuf was transmitted in one or more packets to the PPSN. The port state remains RUNNING.

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25ISC, X25OPC, or X25WIC before executing X25SDM.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- The logical link between the user job and the Gateway node has been disconnected.

19 You have specified an invalid value for dtype.

X25SIM SEND INTERRUPT MESSAGE

Use the X25SIM subroutine to send interrupt data. One execution of X25SIM sends one byte of interrupt data. When you issue an X25SIM subroutine call, the port state must be RUNNING. Only one interrupt can be outstanding at any one time. To find whether an interrupt has been acknowledged, execute the READ PORT STATUS function.

The format of the X25SIM subroutine call is:

```
CALL X25SIM (nport, intbyt, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you set up the virtual circuit by calling X25ISC (INITIATE SWITCHED CIRCUIT), X25OPC (OPEN PERMANENT CIRCUIT), or X25WIC (WAIT INCOMING CALL).

intbyt — is the interrupt byte, an integer in the range 0 to 255.

rcode — is an integer variable in which X25SIM stores a return code. If rcode is nonzero, the subroutine has failed to transmit interrupt data. Rcode values and their meanings are:

0 The X25SIM subroutine is executed successfully: an interrupt message was sent to the PPSN. The port state remains RUNNING.

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X25ISC, X25OPC, or X25WIC before executing X25SIM.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- The logical link between the user job and the Gateway node has been disconnected.

11 Prior to the current execution of X25SIM, you have sent the PPSN an interrupt message that has not yet been confirmed. There can be only one outstanding interrupt at any time.

X25TPA TERMINATE PORT ACCESS

Use the X25TPA subroutine to release all resources associated with a port. Executing the X25TPA subroutine causes an active SVC to be cleared or an active PVC to be released. In either case, the port state changes to UNDEFINED, and any data in transit may be lost. You can issue this subroutine regardless of the current port state.

The format of the X25TPA subroutine is:

```
CALL X25TPA (nport, rcode)
```

where:

nport — is an integer that identifies the port you are using. A value for nport is returned when you set up the virtual circuit by calling X25ISC (INITIATE SWITCHED CIRCUIT), X25OPC (OPEN PERMANENT CIRCUIT), or X25WIC (WAIT INCOMING CALL).

rcode — is an integer variable in which X25TPA stores a return code. Rcode values and their meanings are:

- 0 X25TPA is executed successfully: all resources associated with the port were released, and the port state becomes UNDEFINED.
- 3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs because you have specified a port that has not been allocated. You may have failed to execute X25ISC, X25OPC, or X25WIC before executing X25TPA.

X25WIC WAIT INCOMING CALL

Use the X25WIC subroutine to create a target object to which your X.25 Gateway node can connect when an incoming call arrives from the PPSN. The Network Services Protocol (NSP), which identifies the user process as a passive task, provides the X25WIC subroutine with a network channel. The user process perceives an incoming virtual call as an incoming DECnet logical link from the X.25 Gateway module. You become aware of the incoming call when the port state changes from LISTENING to CALLED.

The format of the X25WIC subroutine call is:

```
CALL X25WIC (name, buffer, nport, rcode)
```

where:

name — is a user-supplied ASCII string that contains the DECnet object identification (in the format of a name or a numeric string) that identifies your process. Your system manager can supply you with the object identification, which is in the X.25 Server Data Base. See the TOPS-10 System Manager's Guide for information on how the X.25 Gateway Software compares data in the X.25 Server Data Base with data in Incoming Call Packets to select which incoming calls should be routed to your process.

buffer — is an integer array that is used as workspace by the X.25 Gateway Access Routines. This buffer can be 144 to 396 words, depending on your selected packet size. To compute the length of the value for buffer, use the following formula:

```
length = 140 + (packet size/4)
```

For example, if your packet size is 16, specify a buffer length of 144.

nport — is an integer that identifies the port you are using. The X25WIC subroutine returns this value, which you must use as an argument for many of the other subroutine calls.

rcode — is an integer variable in which X25WIC stores a return code. If rcode is nonzero, you are not ready to receive an incoming call. Rcode values and their meanings are:

0 X25WIC is executed successfully: a circuit was allocated for receiving incoming calls from the PPSN, the port state changes from UNDEFINED to LISTENING.

2 The TOPS-10 system does not have sufficient resources to allocate a new port.

X25WIC (Cont.)

3 The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has already been allocated.
- A DECnet server logical link could not be established because of system or DECnet errors.

CHAPTER 4

MACRO-10 SUBROUTINE CALLS

4.1 MACRO-10 AND THE TOPS-10 PSI GATEWAY SOFTWARE

The TOPS-10 PSI Gateway Access Routine functions summarized in Chapter 1 are described here as a set of MACRO-10 subroutine calls. To use these subroutine calls, you can write a program that establishes a single communications link on a channel (port). You can then direct program execution by:

- Using the TOPS-10 software interrupt system (see the TOPS-10 Monitor Calls Manual).
- Executing the X%RPS (READ PORT STATUS) function to interrogate the condition of that port.

Before executing a MACRO-10 call, the TOPS-10 PSI Gateway Software stores all registers. After the subroutine has been executed, the Gateway Software restores all registers to their former state and returns control to the main program.

Every MACRO-10 TOPS-10 PSI Gateway subroutine call has a standard form:

- The address of an argument block, ARGBLK, in AC1
- PUSHJ SP, subroutine-address
- A return to the +1 location

Whenever you specify an ASCIZ value for a subroutine argument, supply one of the following:

- A non-null ASCIZ byte pointer that points to a non-null string.
- A non-null ASCIZ byte pointer that points to a null string.
- A binary 0; that is, a null string.

When a subroutine call requires an ASCIZ byte pointer as an argument, you do not need to restrict it to a 7-bit byte pointer. The ASCIZ byte pointer can be of any byte size greater than or equal to 7 bits and up to 36 bits. You must, however, terminate the ASCIZ string with a binary zero byte.

The file UNV:X25SYM.UNV contains universal symbols such as virtual circuit port states, return codes, and other bit-mask symbols. For example, universal symbols for virtual circuit port states are XS%UND and XS%RUN and for return codes are XC%SUC and XC%PER. The user source program must reference this universal file using the SEARCH macro.

Since the TOPS-10 PSI Gateway Access routines use the push down stack for manipulation of data and other tasks, you should initialize the stack pointer to point to a push down list of at least 1000 (octal) words.

On an unsuccessful return from a MACRO-10 call, the following flags are returned in the right half of the return-code word (see Figure 4-1). All bits set to zero indicates success.

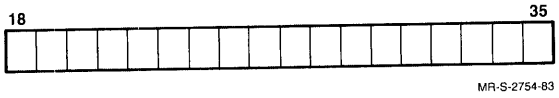


Figure 4-1: Return Code Bit Mask

In all the MACRO-10 function descriptions, each bit in the return code bit mask has a specific meaning. The TOPS-10 system numbers bits from left to right in ascending order. See Table 4-1 for the meaning of each set bit when one or more specific bits of the return code are nonzero.

Table 4-1: Return Code Values

Bit	Symbol	Meaning
18	XC%RSS	Reset seen
19	XC%VSK	Version skew
20	XC%BTS	Buffer is too short
21	XC%BTL	Buffer is too long
22	XC%SDL	Source DTE subaddress is too long
23	XC%DDL	Destination DTE address is too long
24	XC%NIC	No interrupt confirmation is requested
25	XC%AIC	Awaiting interrupt confirmation
26	XC%NDN	No destination
27	XC%INA	Illegal network access code
28	XC%DFT	Data field truncated
29	XC%FFT	Facilities field truncated
30	XC%UNN	Undefined network name
31	XC%NIN	No interrupt to read
32	XC%NDA	No data to read
33	XC%PER	Procedure error
34	XC%IAR	Insufficient access resources
35	XC%IGR	Insufficient gateway resources
None	XC%SUC	Successful

4.2 LINKING A MACRO-10 PROGRAM

The MACRO-10 TOPS-10 PSI Gateway Access Routines are in the library file X25GAM.REL. After you have written and assembled a program to produce relocatable object modules, you must link the modules with X25GAM.REL to produce an executable image file. To link your program, type:

```
.R LINK (RET)
*REL:X25GAM.REL, object-1[, ..., object-n] (RET)
*/SAVE /GO (RET)
```

where:

object-1 through object-n are the names of your relocatable object modules.

4.3 THE SUBROUTINE CALLS

The MACRO-10 calls are in alphabetical order in the following format:

- A title containing the name of the MACRO-10 call and the name of the TOPS-10 PSI Gateway Access Routine function
- The purpose of the function
- A description of the function
- The calling sequence
- The parameter descriptions

In this chapter, all word offsets - such as the number 10 in the expression ARGBLK+10 - are octal numbers. All other numbers in this manual are decimal, unless otherwise noted.

X%AIC ACCEPT INCOMING CALL

Use the X%AIC subroutine to accept an incoming call request. You can use data returned by the X%RIC (READ INCOMING CALL) subroutine to determine whether to accept an incoming call. When you issue an X%AIC subroutine call, the port state must be CALLED.

The calling sequence for the X%AIC subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%AIC
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you called X%WIC (WAIT INCOMING CALL) to set up the virtual circuit.

ARGBLK+1 contains the return code. Following are the X%AIC return codes and their meanings. Upon return from this subroutine, one or more of these can be set:

XC%SUC X%AIC is executed successfully: the incoming call is accepted, and the port state changes from CALLED to RUNNING.

XC%DFT The user-specified accept data field is truncated because it is longer than the limit set by the PPSN.

XC%FFT The facilities data field is truncated because it is longer than the limit set by the PPSN.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%WIC before executing X%AIC.
- You have attempted to execute this subroutine when the port state is not CALLED.
- The logical link between the user job and the Gateway node has been disconnected.

X%AIC (Cont.)

ARGBLK+2 contains an ASCIZ string pointer to the name of the bilateral closed user group or closed user group. This name is user supplied and can be a maximum of 16 ASCII characters.

ARGBLK+3 contains the length and address of the buffer into which you have placed any facilities data required to support your PPSN service. In the left half of ARGBLK+3, specify the length of that buffer in 8-bit bytes. The maximum value you can specify in the left half of ARGBLK+3 depends upon the value you have specified in ARGBLK+2 for this subroutine call:

- If you specify a bilateral closed user group in ARGBLK+2, the maximum value you can specify in the left half of ARGBLK+3 is 60.
- If you specify a closed user group in ARGBLK+2, the maximum value you can specify in the left half of ARGBLK+3 is 61.
- If you do not specify a value in ARGBLK+2, the maximum value you can specify in the left half of ARGBLK+3 is 63.

Your PPSN may further restrict this value. In the right half of ARGBLK+3, specify the address of the buffer. (For information on facilities data, see the documentation for your PPSN.)

ARGBLK+4 contains the length and address of the buffer into which you have placed any user-specified accept data. In the left half of ARGBLK+4, specify the length of that buffer in 8-bit bytes. The maximum length normally is 16 8-bit bytes (or 128 for fast select calls), although your PPSN may further restrict this value. In the right half of ARGBLK+4, specify the address of the buffer.

X%CIM CONFIRM INTERRUPT MESSAGE

Use the X%CIM subroutine to confirm the receipt of interrupt data. When you issue an X%CIM subroutine call, the port state must be RUNNING. Only one interrupt can be outstanding at a time.

The calling sequence for the X%CIM subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%CIM
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you set up the virtual circuit by calling X%ISC (INITIATE SWITCHED CIRCUIT) X%OPC (OPEN PERMANENT CIRCUIT) or X%WIC (WAIT INCOMING CALL).

ARGBLK+1 contains the return code. Following are the X%CIM return codes and their meanings. Upon return from this subroutine, one or more of these may be set:

XC%SUC X%CIM is executed successfully: the interrupt is confirmed. The port state remains RUNNING.

XC%NIC No interrupt confirmation is requested. The user has not received an interrupt message from the PPSN that requires a confirmation.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%ISC, X%OPC, or X%WIC before executing X%CIM.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- The logical link between the user job and the Gateway node has been disconnected.

X%CSC CLEAR SWITCHED CIRCUIT

Use the X%CSC subroutine to terminate an SVC connection. Once you have executed the X%CSC subroutine, any data in transit may be lost, and you can no longer communicate over the specified circuit. However, the port remains assigned; to release the port, execute the X%TPA (TERMINATE PORT ACCESS) subroutine. When you issue an X%CSC subroutine call, the port state must be CALLED, CALLING, RUNNING, SYNC, or UNSYNC.

The calling sequence for the X%CSC subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%CSC
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you called X%ISC (INITIATE SWITCHED CIRCUIT) or X%WIC (WAIT INCOMING CALL) to set up the virtual circuit.

ARGBLK+1 contains the return code. Following are the X%CSC return codes and their meanings. Upon return from this subroutine, one or more of these may be set:

XC%SUC X%CSC is executed successfully: a clear request is sent to the PPSN, and the port state changes from RUNNING to CLEARING.

XC%DFT The user-specified clear data field is truncated because it is longer than the limit set by the PPSN.

XC%FFT The facilities data field is truncated because it is longer than the limit set by the PPSN.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%ISC or X%WIC before executing X%CSC.
- You have attempted to execute this subroutine when the port state is not CALLING, CALLED, RUNNING, SYNC, or UNSYNC.
- The logical link between the user job and the Gateway node has been disconnected.

ARGBLK+2 contains the DTE clear cause. The X.25 software passes the rightmost 8-bit byte to the remote DTE.

X%CSC (Cont.)

ARGBLK+3 contains an ASCIZ byte pointer to the name of the bilateral closed user group or closed user group. This name is user supplied and can be a maximum of 16 ASCII characters. Some PPSNs may not allow this field.

ARGBLK+4 contains the length and address of the buffer into which you have placed any facilities data required to support your PPSN service. In the left half of ARGBLK+4, specify the length of that buffer in 8-bit bytes. The maximum value you can specify in the left half of ARGBLK+4 depends upon the value you have specified in ARGBLK+3 for this subroutine call:

- If you specify a bilateral closed user group in ARGBLK+3, the maximum value you can specify in the left half of ARGBLK+4 is 60.
- If you specify a closed user group in ARGBLK+3, the maximum value you can specify in the left half of ARGBLK+4 is 61.
- If you do not specify a value in ARGBLK+3, the maximum value you can specify in the left half of ARGBLK+4 is 63.

Your PPSN may further restrict this value. In the right half of ARGBLK+4, specify the address of the buffer. (For information on facilities data, see the documentation for your PPSN.)

ARGBLK+5 contains the length and address of the buffer into which you have placed any user-specified clear data. In the left half of ARGBLK+5, specify the length of that buffer in 8-bit bytes. The maximum length normally is 16 8 bit bytes (or 128 for fast select calls), although your PPSN may further restrict this value. In the right half of ARGBLK+5, specify the address of the buffer.

X%ISC INITIATE SWITCHED CIRCUIT

Use the X%ISC subroutine to issue a virtual call request over a switched virtual circuit to a remote DTE. A successful return does not indicate that the remote DTE has responded to the call. When the remote DTE responds to the call, the port state changes from CALLING to RUNNING or CLEARED. When you issue an X%ISC subroutine call, the port state must be UNDEFINED. (Note that the remote DTE is also called the destination DTE, and the local DTE can be called the source DTE.)

The calling sequence for the X%ISC subroutine call is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%ISC
```

where:

ARGBLK+0 contains the software interrupt channel and the PRTBLK. In the left half of ARGBLK+0, specify a positive number to indicate software interrupt service to be enabled for the logical link to the X.25 node. In the right half of ARGBLK+0, specify the address of a block of storage called PRTBLK, which the X.25 Gateway Access Routines use as a workspace. To compute the length of PRTBLK in words, use the following formula:

length = 140 + (packet size/4)

For example, if your packet size is 16, specify a PRTBLK length of 144.

The X.25 software returns the port number in ARGBLK+0. You must use this value when calling many of the other subroutines. The port number is also the interrupt channel when you specify a positive number in the left half for input. You can use it to determine which channel the software interrupt has been granted by the system.

ARGBLK+1 contains the return code. Following are the X%ISC return codes and their meanings. Upon return from this subroutine, one or more of these can be set:

XC%SUC X%ISC is executed successfully: a call request is sent to the PPSN, and the port state changes from UNDEFINED to CALLING.

XC%VSK The version numbers of the TOPS-10 PSI Gateway Access Software and the TOPS-10 PSI Gateway Software are not compatible.

XC%SDL The specified source DTE subaddress is too long.

XC%DDL The specified destination DTE address is too long.

XC%NDN You have not specified the destination DTE address to which the circuit is to be established.

X%ISC (Cont.)

XC%INA You have not specified a password or you have specified an incorrect password for gaining access to the PPSN through the Gateway node.

XC%DFT The user-specified call data field is truncated because it is longer than the limit set by the PPSN.

XC%FFT The facilities data field is truncated because it is longer than the limit set by the PPSN.

XC%UNN You have specified a network name that has not been defined in the DECnet-10 Network Management data base.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port which has already been allocated.
- A logical link could not be established to the Gateway node for reasons other than those specified here.

XC%IAR The TOPS-10 PSI Gateway Access Routines do not have sufficient resources to allocate a new circuit.

XC%IGR The TOPS-10 PSI Gateway does not have sufficient resources to allocate a new switched circuit.

ARGBLK+2 contains an ASCIZ byte pointer to the name of the PPSN over which you want to communicate. This name is user supplied and can be a maximum of 16 ASCII characters.

ARGBLK+3 contains an ASCIZ byte pointer to the password required for gaining access to the X.25 Gateway. This password is user supplied and can be a maximum of 16 ASCII characters. Your system manager determines this password.

ARGBLK+4 contains an ASCIZ byte pointer to the address of the destination DTE. This address is user supplied and can be a maximum of 15 ASCII characters.

ARGBLK+5 contains an ASCIZ byte pointer to the subaddress of the source DTE. This subaddress is user supplied and can be a maximum of 15 ASCII characters.

X%ISC (Cont.)

ARGBLK+6 contains an ASCIZ byte pointer to the name of the bilateral closed user group or closed user group. This name is user supplied and can be a maximum of 16 ASCII characters.

ARGBLK+7 contains the length and address of the buffer into which you have placed any facilities data required to support your PPSN service. In the left half of ARGBLK+7, specify the length of that buffer in 8-bit bytes. The maximum value you can specify in the left half of ARGBLK+7 depends upon the value you have specified in ARGBLK+6 for this subroutine call:

- If you specify a bilateral closed user group in ARGBLK+6, the maximum value you can specify in the left half of ARGBLK+7 is 60.
- If you specify a closed user group in ARGBLK+6, the maximum value you can specify in the left half of ARGBLK+7 is 61.
- If you do not specify a value in ARGBLK+6, the maximum value you can specify in the left half of ARGBLK+7 is 63.

Your PPSN may further restrict this value. In the right half of ARGBLK+7, specify the address of the buffer. (For information on facilities data, see the documentation for your PPSN.)

ARGBLK+10 contains the length and address of the buffer into which you have placed any user-specified call data. In the left half of ARGBLK+8, specify the length of that buffer in 8-bit bytes. The maximum length normally is 16 8-bit bytes (or 128 for fast select calls), although your PPSN may further restrict this value. In the right half of ARGBLK+8, specify the address of the buffer.

X%NCS NO COMMUNICATION SEEN

Use the X%NCS subroutine to acknowledge one or more failures of the permanent virtual circuit due to loss of communication with the PPSN. After such a failure, you retain possession of the permanent virtual circuit. When you issue an X%NCS call, the port state must be NO COMMUNICATION.

The calling sequence for the X%NCS subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%NCS
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you called the subroutine X%OPC (OPEN PERMANENT CIRCUIT) to set up the virtual circuit.

ARGBLK+1 is the return code. Following are the X%NCS return codes and their meanings:

XC%SUC The X%NCS subroutine is executed successfully: you have acknowledged the loss of communication with the PPSN and are using the permanent virtual circuit, and the port state changes from NO COMMUNICATION to RUNNING.

XC%PER The TOPS-10 PSI Gateway Access Software encounters a procedure error as it tried to execute this subroutine. The procedure error occurred for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%OPC before executing X%NCS.
- You have attempted to execute this subroutine when the port state is not NO COMMUNICATION.
- You have attempted to execute this subroutine with a switched virtual circuit.
- The logical link between the user job and the Gateway node has been disconnected.

X%OPC OPEN PERMANENT CIRCUIT

Use the X%OPC subroutine to acquire exclusive use of a specified PVC. The function does not try to contact the remote DTE. A successful response changes the port state to RUNNING or UNSYNC. You can transfer data over the PVC as soon as the port state is RUNNING.

A port state of UNSYNC indicates that the remote DTE has issued a reset; you must issue an X%RVC (RESET VIRTUAL CIRCUIT) subroutine call before you can proceed. When you issue an X%OPC subroutine call, the port state must be UNDEFINED.

The calling sequence for the X%OPC subroutine call is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%OPC
```

where:

ARGBLK+0 contains the software interrupt channel and the PRTBLK. In the left half of ARGBLK+0, specify a positive number to indicate software interrupt service to be enabled for the the logical link to the X.25 node. In the right half of ARGBLK+0, specify the address of a block of storage called PRTBLK, which the X.25 Gateway Access Routines use as a workspace. To compute the length of PRTBLK in words, use the following formula:

length = 140 + (packet size/4)

Therefore, if your packet size is 16, for example, specify a PRTBLK length of 144.

The X.25 software returns the port number in ARGBLK+0. You must use this value when calling many of the other subroutines. The port number is also the interrupt channel when you specify a positive number in the left half for input. You can use it to determine which channel the software interrupt has been granted by the system.

ARGBLK+1 contains the return code. Following are the X%OPC return codes. Upon return from this subroutine, one or more of these can be set:

XC%SUC X%OPC is executed successfully: a permanent virtual circuit is allocated exclusively for the user job, and the port state changes from UNDEFINED to OPEN.

XC%VSK The version numbers of the TOPS-10 PSI Gateway Access Routines and the TOPS-10 PSI Gateway Software are not compatible.

XC%NDN You have specified a permanent circuit that has not been defined in the DECnet-10 Network Management data base.

XC%INA You have not specified a password, or you have specified an incorrect password for gaining access to the PPSN through the Gateway node.

X%OPC (Cont.)

XC%UNN You have specified a network name that has not been defined in the DECnet-10 Network Management data base.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has already been allocated.
- A logical link could not be established to the Gateway node for reasons other than those specified above.

XC%IAR The TOPS-10 system does not have sufficient resources to allocate a new circuit.

XC%IGR The TOPS-10 PSI Gateway does not have sufficient resources to allocate a new permanent circuit.

ARGBLK+2 contains an ASCIZ byte pointer to the name of the PPSN over which you want to communicate. This name is user supplied and can be a maximum of 16 ASCII characters.

ARGBLK+3 contains an ASCIZ byte pointer to the password required for gaining access to the X.25 Gateway. This password is user supplied and can be a maximum of 16 ASCII characters. Your system manager determines this password.

ARGBLK+4 contains an ASCIZ byte pointer to the name of the PVC you are using. This name is user supplied and can be a maximum of 16 ASCII characters. Your system manager can supply you with the PVC name.

X%RAD READ ACCEPT DATA

Use the X%RAD subroutine to read information obtained from the remote DTE after it accepted your request to establish a switched virtual circuit connection. Some PPSNs do not supply data in all the described fields. Where no data is supplied, the fields are filled with nulls (binary 0s) or the corresponding length field is zero. When you issue an X%RAD subroutine call, the port state must be RUNNING.

The calling sequence for the X%RAD subroutine call is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%RAD
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you call X%ISC (INITIATE SWITCHED CIRCUIT) to set up the virtual circuit.

ARGBLK+1 contains the return code. Following are the X%RAD return codes and their meanings. Upon return from this subroutine, one or more of these can be set:

XC%SUC X%RAD is executed successfully: the user accept data and/or facilities data is returned. The port state remains RUNNING.

XC%DFT The accept data field is truncated because you did not supply a buffer large enough to contain that data.

XC%FFT The facilities data field is truncated because you did not supply a buffer large enough to contain that data.

XC%NDA There is no user accept data.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%ISC before executing X%RAD.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- You have attempted to execute this subroutine with a permanent virtual circuit.
- The logical link between the user job and the Gateway node has been disconnected.

X%RAD (Cont.)

ARGBLK+2 contains an ASCIZ byte pointer to the buffer in which X%RAD returns the binary closed user group or closed user group supplied by the remote DTE. The group name can be a maximum of 16 ASCII characters.

ARGBLK+3 contains the length and address of the buffer in which X%RAD returns facilities data supplied by the remote DTE. In the left half of ARGBLK+3, specify the maximum number of 8-bit bytes that the buffer can receive. This data can be as many as 63 8-bit bytes. Upon return, X%RAD sets the left half of ARGBLK+3 equal to the actual number of 8-bit bytes returned in the buffer. In the right half of ARGBLK+3, specify the address of this buffer. (For information on facilities data, see the documentation for your PPSN.)

ARGBLK+4 contains the length and address of the buffer in which X%RAD returns accept data supplied by the remote DTE. In the left half of ARGBLK+4, specify the maximum number of 8-bit bytes that the buffer can receive. This data can be as many as 16 8-bit bytes (or 128 for fast select calls). Upon return, X%RAD sets the left half of ARGBLK+4 equal to the actual number of 8-bit bytes returned in the buffer. In the right half of ARGBLK+4, specify the address of this buffer.

X%RCD READ CLEAR DATA

Use the X%RCD subroutine to read data obtained when an SVC is cleared. The content of the data is PPSN specific, but not all PPSNs provide this information when an SVC is cleared. Use the X%RPS (READ PORT STATUS) subroutine to determine that the PPSN has cleared the virtual circuit. When you issue an X%RCD subroutine call, the port state must be CLEARED.

The calling sequence for the X%RCD subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%RCD
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you call X%ISC (INITIATE SWITCHED CIRCUIT) or X%WIC (WAIT INCOMING CALL) to set up the virtual circuit.

ARGBLK+1 contains the return code. Following are the X%RCD return codes and their meanings. Upon return from this subroutine, one or more of these can be set:

XC%SUC X%RCD is executed successfully: the clear cause, clear diagnostic, and/or user clear data, and/or facilities are returned. The port state remains CLEARED.

XC%DFT The user clear data field is truncated because you did not supply a buffer large enough to contain that data.

XC%FFT The facilities data field is truncated because you did not supply a buffer large enough to contain that data.

XC%NDA There is no user clear data.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%ISC or X%WIC before executing X%RCD.
- You have attempted to execute this subroutine when the port state is not CLEARED.

X%RCD (Cont.)

- You have attempted to execute this subroutine with a permanent virtual circuit.
- The logical link between the user job and the Gateway node has been disconnected.

ARGBLK+2 contains the clear cause bytes returned by X%RCD. In the left half of ARGBLK+2, X%RCD returns the network clear cause supplied by the PPSN. If this value is nonzero, the network has cleared the call. This value is the network clear cause and the network clear diagnostic is in the right half of this word. If this value is zero, the remote user has cleared the SVC, and the user clear diagnostic is contained in the right half of ARGBLK+2. (For information on the network clear cause and diagnostic, see the documentation for your PPSN.)

ARGBLK+3 contains an ASCIZ byte pointer to the buffer in which X%RCD returns the binary closed user group or closed user group supplied by the remote DTE. The group name can be a maximum of 16 ASCII characters.

ARGBLK+4 contains the length and address of the buffer in which X%RCD returns facilities data supplied by the remote DTE. In the left half of ARGBLK+4, specify the maximum number of 8-bit bytes that the buffer can receive. This data can be as many as 63 8-bit bytes. Upon return, X%RCD sets the left half of ARGBLK+4 equal to the actual number of 8-bit bytes returned in the buffer. In the right half of ARGBLK+4, specify the address of this buffer. (For information on facilities data, see the documentation for your PPSN.)

ARGBLK+5 contains the length and address of the buffer in which X%RCD returns clear data supplied by the remote DTE. In the left half of ARGBLK+5, specify the maximum number of 8-bit bytes that the buffer can receive. This data can be as many as 16 8-bit bytes (or 128 for fast select calls). Upon return, X%RCD sets the left half of ARGBLK+5 to the actual number of 8-bit bytes returned in the buffer. In the right half of ARGBLK+4, specify the address of this buffer.

X%RDM READ DATA MESSAGE

Use the X%RDM subroutine to receive data from a virtual circuit. The data can be either normal or qualified. When you issue an X%RDM subroutine call, the port state must be RUNNING.

The calling sequence for the X%RDM subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%RDM
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you set up the virtual circuit by calling X%ISC (INITIATE SWITCHED CIRCUIT), X%OPC (OPEN PERMANENT CIRCUIT), or X%WIC (WAIT INCOMING CALL).

ARGBLK+1 contains the return code. Following are the X%RDM return codes and their meanings. Upon return from this subroutine, one or more of these can be set:

XC%SUC X%RDM is executed successfully: a normal or qualified user data packet is returned. The port state remains RUNNING.

XC%RSS There is no data from the PPSN to be returned because the circuit has been reset.

XC%BTS The buffer is too short. You receive a partially filled data packet with the more bit set. The more bit is set upon return from the function. However, it is entirely up to you how you handle such a data packet. For example, you can choose to ignore the data packet and reset the circuit.

XC%DFT The data field is truncated because you supplied a buffer which is not large enough to accommodate the entire data packet.

XC%NDA There is no user data to be returned.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%ISC, X%OPC, or X%WIC before executing X%RDM.
- You have attempted to execute this subroutine when the port state is not RUNNING.

X%RDM (Cont.)

- The logical link between the user job and the Gateway node has been disconnected.

ARGBLK+2 contains the more-data and data-qualifier bits (see Table 4-2) and the number of bytes of data received. Interpret the received more-data and data-qualifier bits in the following way:

- If bit 0 is 0, the more-data bit is not set. This means that the next packet is not logically related to the current one.
- If bit 0 is 1, the more-data bit is set. This means that the packet you have just read is full and the next packet is logically related to the current one (the nature of this relationship is user defined).
- If bit 1 is 0, the data you have just read is normal data.
- If bit 1 is 1, the data you have just read is qualified data.

In the right half of ARGBLK+2, specify the maximum number of bytes that you can receive. Upon return, X%RDM sets ARGBLK+2 to the actual number of bytes received. If the actual number of bytes received is 0 and the return code in ARGBLK+1 is XC%SUC (successful), you have received an empty Data packet.

ARGBLK+3 contains the destination byte pointer to the location in which X%RDM returns incoming data. Upon return, X%RDM sets this byte pointer to the first byte following the last received byte.

Table 4-2: More-bit and Data-qualifier Bit Meanings

Bit	Symbol	Meaning
0	XM%MOR	More bit
1	XM%QUA	Qualified data bit

X%RIC READ INCOMING CALL

Use the X%RIC subroutine to obtain information about an incoming call when the port state changes from LISTENING to CALLED. When you issue an X%RIC subroutine call, the port state must be CALLED.

The calling sequence for the X%RIC subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%RIC
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you call X%WIC (WAIT INCOMING CALL) to set up the virtual circuit.

ARGBLK+1 contains the return code. Following are the X%RIC return codes and their meanings. Upon return from this subroutine, one or more of these can be set:

XC%SUC X%RIC is executed successfully: incoming call data and/or call facilities are returned. The port state remains CALLED.

XC%DFT The user-specified call data field is truncated because you did not supply a buffer large enough to contain that data.

XC%FFT The facilities field is truncated because you did not supply a buffer large enough to contain that data.

XC%NDA There is no user call data.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%WIC before executing X%RIC.
- You have attempted to execute this subroutine when the port state is not CALLED.
- The logical link between the user job and the Gateway node has been disconnected.

ARGBLK+2 contains an ASCIZ byte pointer to the buffer in which X%RIC returns the name of the network from which the call originated. This name can be a maximum of 16 ASCII characters.

X%RIC (Cont.)

ARGBLK+3	contains an ASCIZ byte pointer to the buffer in which X%RIC returns the address of the calling (remote) DTE. This address can be a maximum of 16 ASCII characters.
ARGBLK+4	contains an ASCIZ byte pointer to the buffer in which X%RIC returns the subaddress of the called (local) DTE. This subaddress can be a maximum of 16 ASCII characters.
ARGBLK+5	contains an ASCIZ byte pointer to the buffer in which X%RIC returns the binary closed user group or closed user group supplied by the remote DTE. The group name can be a maximum of 16 ASCII characters.
ARGBLK+6	contains the length and address of the buffer in which X%RIC returns facilities data supplied by the remote DTE. In the left half of ARGBLK+6, specify the maximum number of 8-bit bytes that the buffer can receive. This data can be as many as 63 8-bit bytes. Upon return, X%RIC sets the left half of ARGBLK+6 equal to the actual number of 8-bit bytes returned in the buffer. In the right half of ARGBLK+6, specify the address of this buffer. (For information on facilities data, see the documentation for your PPSN.)
ARGBLK+7	contains the length and address of the buffer in which X%RIC returns user-specified call data supplied by the remote DTE. In the left half of ARGBLK+7, specify the maximum number of 8-bit bytes that the buffer can receive. This data can be as many as 16 8-bit bytes (or 128 for fast select calls). Upon return, X%RIC sets the left half of ARGBLK+7 equal to the actual number of 8-bit bytes returned in the buffer. In the right half of ARGBLK+7, specify the address of this buffer.

X%RIM READ INTERRUPT MESSAGE

Use the X%RIM subroutine to receive interrupt data. When you issue an X%RIM subroutine call, the port state must be RUNNING. Use the X%RPS (READ PORT STATUS) subroutine to determine whether there is an interrupt byte to read.

The calling sequence for the X%RIM subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%RIM
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you set up the virtual circuit by calling X%ISC (INITIATE SWITCHED CIRCUIT), X%OPC (OPEN PERMANENT CIRCUIT, or X%WIC (WAIT INCOMING CALL).

ARGBLK+1 contains the return code. Following are the X%RIM return codes and their meanings. Upon return from this subroutine, one or more of these can be set:

XC%SUC X%RIM is executed successfully: an interrupt data byte is returned. The port state remains RUNNING.

XC%RSS There is no interrupt from the PPSN to be returned because the circuit has been reset.

XC%NIN There is no interrupt.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%ISC, X%OPC, or X%WIC before executing X%RIM.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- The logical link between the user job and the Gateway node has been disconnected.

ARGBLK+2 contains a word in which X%RIM returns the interrupt byte.

X%RPS READ PORT STATUS

Use the X%RPS subroutine to poll the port maintained by the TOPS-10 PSI Gateway Software. You can issue an X%RPS subroutine call regardless of the current port state. However, if the port state is UNDEFINED, the left half of ARGBLK+2 and all of ARGBLK+3 have indeterminate meaning.

The calling sequence for the X%RPS subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%RPS
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you set up the virtual circuit by calling X%ISC (INITIATE SWITCHED CIRCUIT), X%OPC (OPEN PERMANENT CIRCUIT), or X%WIC (WAIT INCOMING CALL).

ARGBLK+1 contains the return code. Following are the X%RPS return codes and their meanings:

- XC%SUC X%RPS is executed successfully: the port status and flags are returned. The port state remains unchanges.
- XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs because the logical link between the user job and the Gateway node has been disconnected.

ARGBLK+2 contains the error condition and port state. In the left half of ARGBLK+2, X%RPS returns the last fatal error condition (see Table 4-3) that changes the port state to the ERROR state. In the right half of ARGBLK+2, X%RPS returns the current port state. See Tables 2-3 and 4-4 for information about port states.

X%RPS (Cont.)

ARGBLK+3 contains the interrupt/data bit mask (see Table 4-5) and the packet size. In the left half of ARGBLK+3, X%RPS sets bits that indicate the following:

- If bit 0 is set, there is an incoming interrupt to be confirmed. Use X%CIM (CONFIRM INTERRUPT MESSAGE) to confirm the interrupt.
- If bit 1 is set, an outgoing interrupt has been transmitted but not confirmed by the remote DTE.
- If bit 2 is set, incoming data is available on the data subchannel. Use X%RDM (READ DATA MESSAGE) to read the data.
- If bit 3 is set, incoming interrupt data is available on the interrupt subchannel. Use X25RIM (READ INTERRUPT MESSAGE) to read the data.

In the right half of ARGBLK+3, X%RPS returns the current packet size.

Table 4-3: Fatal Error Conditions

Bit	Symbol	Meaning
6	XE%IUG	Invalid user group
7	XE%VSK	Version skew
8	XE%DDL	Destination DTE address is too long
9	XE%SDL	Source DTE subaddress is too long
10	XE%FFT	Facilities field truncated
11	XE%DFT	Data field truncated
12	XE%NCM	No communication with the Public Network
13	XE%UNN	Undefined network name
14	XE%UCN	Undefined circuit name
15	XE%INU	Circuit in use (reserved)
16	XE%IGR	Insufficient gateway resources
17	XE%UNK	Unknown error

X%RPS (Cont.)

Table 4-4: Port States

Decimal Value	Symbol	Meaning
0	XS%UND	UNDEFINED
1	XS%OPN	OPEN
2	XS%CAG	CALLING
3	XS%LSN	LISTENING
4	XS%CAD	CALLED
5	XS%RUN	RUNNING
6	XS%SYN	SYNC
7	XS%UNS	UNSYNC
8	XS%CLG	CLEARING
9	XS%CLD	CLEARED
10	XS%ERR	ERROR
11	XS%NCM	NO COMMUNICATION

Table 4-5: Interrupt/Data Bit Mask Meanings

Bit	Symbol	Meaning
0	XM%IIC	Incoming interrupt confirmation pending
1	XM%OIC	Outgoing interrupt confirmation pending
2	XM%DAT	Incoming data is available
3	XM%INT	Incoming interrupt data is available

X%RRD READ RESET DATA

Use the X%RRD subroutine to read information obtained when a virtual circuit is reset. The content of the data is network specified, but not all networks provide this information when a call is reset. Use the X%RPS (READ PORT STATUS) subroutine to determine that the virtual circuit has been reset. When you issue an X%RRD subroutine call, the port state must be UNSYNC.

The calling sequence for the X%RRD subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%RRD
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you set up the virtual circuit by calling X%ISC (INITIATE SWITCHED CIRCUIT), X%OPC (OPEN PERMANENT CIRCUIT), or X%WIC (WAIT INCOMING CALL).

ARGBLK+1 contains the return code. Following are the X%RRD return codes and their meanings:

XC%SUC X%RRD is executed successfully: the reset cause and reset diagnostic bytes are returned. The port state remains UNSYNC.

XC%PER The TOPS-10 PSI Gateway software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%ISC, X%OPC, or X%WIC before executing X%RRD.
- You have attempted to execute this subroutine when the port state is not UNSYNC.
- The logical link between the user job and the Gateway node has been disconnected.

ARGBLK+2 contains the reset cause and diagnostic bytes. In the left half of ARGBLK+2, X%RRD returns the network reset cause byte supplied by the PPSN. If this value is nonzero, the network has reset the virtual circuit, and this value is the network reset cause and the network diagnostic is contained in the right half of this word. If this value is zero, the remote user has reset the virtual circuit, and the user reset diagnostic byte is contained in the right half of ARGBLK+2. (For information on the network reset cause byte, see the documentation for your PPSN.)

X%RVC RESET VIRTUAL CIRCUIT

Use the X%RVC subroutine to reinitialize a virtual circuit and return it to the state it was in before data is transferred over that circuit. At the time of resetting, the PPSN might discard data and interrupt packets that are in transit. Also, use the X%RVC subroutine to acknowledge a reset from the network.

When you initiate a reset, the port state changes from RUNNING to SYNC. When the PPSN initiates a reset, the port state changes from RUNNING to UNSYNC. Acknowledge a PPSN reset as soon as possible because the network may time out and clear the circuit before you acknowledge. When you issue an X%RVC subroutine call, the port state must be either RUNNING or UNSYNC.

The calling sequence for the X%RVC subroutine call is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%RVC
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you set up the virtual circuit by calling X%ISC (INITIATE SWITCHED CIRCUIT), X%OPC (OPEN PERMANENT CIRCUIT), or X%WIC (WAIT INCOMING CALL).

ARGBLK+1 contains the return code. The X%RVC return codes and their meanings are:

XC%SUC X%RVC is executed successfully. If you use X%RVC to initiate a reset on the circuit, a reset request is sent to the PPSN and the port state changes from RUNNING to SYNC. If you use X%RVC to confirm a reset indication from the PPSN, a reset confirmation is sent to the PPSN, and the port state changes from UNSYNC to RUNNING.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%ISC, X%OPC, or X%WIC before executing X%RVC.
- You have attempted to execute this subroutine when the port state is not RUNNING or UNSYNC.
- The logical link between the user job and the Gateway node has been disconnected.

X%RVC (Cont.)

ARGBLK+2	contains an 8-bit byte in which you supply user-defined diagnostic data to the remote user. The X.25 software ignores this value if you use X%RVC to confirm a network reset.

X%SDM SEND DATA MESSAGE

Use the X%SDM subroutine to queue a buffer of data for transmission to the PPSN. If this subroutine is executed successfully, the X.25 Gateway Access Routines have successfully transmitted the buffer. However, successful execution does not mean that the data has reached the PPSN.

Normally, the X%SDM subroutine is blocked until the entire data message is transmitted successfully to the TOPS-10 PSI Gateway. If your program needs to perform non-blocking I/O functions, you must check the status of the channel (and, therefore, the link to the TOPS-10 PSI Gateway) prior to calling the X%SDM subroutine as follows:

```
        .
        .
ARGBLK: BLOCK   ^D10                        ;X.25 argument block
RSBLK:  XWD     <(NS.WAI)>+.NSFRS,2         ;Wait bit, function, block length
        XWD     0,0                         ;Code supplies channel number
        .
        .
        MOVE    S1,ARGBLK                   ;Get port number (also channel number)
        HRRM    S1,RSBLK+.NSACH             ;Store channel number in argument block
        MOVEI   S1,RSBLK
        NSP.    S1,                         ;Read status
         JRST   ERROR                       ;Error
        MOVEI   S1,NS.NDR                   ;Set up bit mask
        TDNN    S1,RSBLK+.NASCH             ;Is channel free for output?
         JRST   LSWARE                      ;No, go elsewhere
        MOVEI   S1,ARGBLK
        CALL    X%SDM##                     ;Send the string to the network
        .
        .
```

With X%SDM, you can send data over either a normal or a qualified data subchannel.

When you issue an X%SDM subroutine call, the port state must be RUNNING.

The calling sequence for the X%SDM subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%SDM
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you set up the virtual circuit by calling X%ISC (INITIATE SWITCHED CIRCUIT), X%OPC (OPEN PERMANENT CIRCUIT), or X%WIC (WAIT INCOMING CALL).

ARGBLK+1 contains the return code. Following are the X%SDM return codes and their meanings. Upon return from this subroutine, one or more of these can be set:

XC%SUC X%SDM is executed successfully: a data packet is sent to the PPSN. The port state remains RUNNING.

X%SDM (Cont.)

XC%BTS You attempt to send a buffer of data with the more bit set, but the buffer is smaller than the packet size that has been agreed upon by the user job and the PPSN during circuit initialization time.

XC%BTL You attempt to send a buffer of data which is larger than the packet size that has been agreed upon by the user job and the PPSN during circuit initialization time.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%ISC, X%OPC, or X%WIC before executing X%SDM.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- The logical link between the user job and the Gateway node has been disconnected.

ARGBLK+2 contains the more-data and data-qualifier bits (see Table 4-2) and the length of data to be transmitted. Set the following bits for the following purposes:

- To set the more-data bit, set bit 0 to 1. You can set the more-data bit only if you are sending a full packet. Setting the more data bit also implies that the next packet is logically related to the current one (the nature of this relationship is user defined).
- If you are transmitting normal data, set bit 1 to 0. If you are transmitting qualified data, set bit 1 to 1.

In the right half of ARGBLK+2, specify the length in bytes of the data to be transmitted.

ARGBLK+3 contains the source byte pointer of the location of the data to be transmitted. The data must be in bytes of 8 bits or less. Upon return, X%SDM sets ARGBLK+3 to the first byte following the last transmitted byte.

X%SIM SEND INTERRUPT MESSAGE

Use the X%SIM subroutine to send interrupt data. One execution of X%SIM sends one byte of interrupt data. When you issue an X%SIM subroutine call, the port state must be RUNNING. Only one interrupt can be outstanding at any one time. To determine if an interrupt has been acknowledged, execute the READ PORT STATUS function.

The calling sequence for the X%SIM subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%SIM
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you set up the virtual circuit by calling X%ISC (INITIATE SWITCHED CIRCUIT), X%OPC (OPEN PERMANENT CIRCUIT), or X%WIC (WAIT INCOMING CALL).

ARGBLK+1 is the return code. Following are the X%SIM return codes and their meanings. Upon return from this subroutine, one or more of these can be set:

XC%SUC X%SIM is executed successfully: an interrupt message is sent to the PPSN. The port state remains RUNNING.

XC%AIC Prior to this execution of X%SIM, you have sent the PPSN an interrupt message that has not yet been confirmed. There can be only one outstanding interrupt at any time.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has not been allocated. You may have failed to execute X%ISC, X%OPC, or X%WIC before executing X%SIM.
- You have attempted to execute this subroutine when the port state is not RUNNING.
- The logical link between the user job and the Gateway node has been disconnected.

ARGBLK+2 contains one 8-bit byte of interrupt data.

X%TPA TERMINATE PORT ACCESS

Use the X%TPA subroutine to release all resources associated with a port. Executing the X%TPA subroutine causes either an active SVC to be cleared or an active PVC to be released. In either case, the port state changes to UNDEFINED, and any data in transit can be lost. You can issue this subroutine regardless of the current port state.

The calling sequence for the X%TPA subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%TPA
```

where:

ARGBLK+0 contains the port number, which identifies the port you are using. The port number is returned when you set up the virtual circuit by calling X%ISC (INITIATE SWITCHED CIRCUIT), X%OPC (OPEN PERMANENT CIRCUIT), or X%WIC (WAIT INCOMING CALL).

ARGBLK+1 contains the return code. Following are the X%TPA return codes and their meanings:

XC%SUC X%TPA is executed successfully: all resources associated with the port are released and the port state became UNDEFINED.

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs because you have specified a port that has not been allocated. You may have failed to execute X%ISC, X%OPC, or X%WIC before executing X%TPA.

X%WIC WAIT INCOMING CALL

Use the X%WIC subroutine to create a target object to which your X.25 Gateway node can connect when an incoming call arrives from the PPSN. The X%WIC subroutine obtains a network channel from Network Services Protocol (NSP) that identifies the user process as a passive task. The user process perceives an incoming virtual call as an incoming DECnet logical link from the X.25 Gateway module. You perceive the incoming call when the port state changes from LISTENING to CALLED.

The calling sequence for the X%WIC subroutine is:

```
MOVEI AC1,ARGBLK
PUSHJ P,X%WIC
```

where:

ARGBLK+0 contains the software interrupt channel and the PRTBLK. In the left half of ARGBLK+0, specify a positive number to indicate software interrupt service to be enabled for the logical link to the X.25 node. In the right half of ARGBLK+0, specify the address of a block of storage called PRTBLK, which the X.25 Gateway Access Routines use as a workspace. To compute the length of PRTBLK in words, use the following formula:

length = 140 + (packet size/4)

Therefore, if your packet size is 16, for example, specify a PRTBLK length of 144.

The X.25 software returns the port number in ARGBLK+0. You must use this value when calling many of the other subroutines. The port number is also the interrupt channel when you specify a positive number in the left half for input. You can use it to determine which channel the software interrupt has been granted by the system.

ARGBLK+1 contains the return code. Following are the X%WIC return codes and their meanings:

XC%SUC X%WIC is executed successfully: a circuit is allocated to receive incoming calls from the PPSN, and the port state changes from UNDEFINED to LISTENING.

XC%IAR The TOPS-10 system does not have sufficient resources to allocate a new port.

X%WIC (Cont.)

XC%PER The TOPS-10 PSI Gateway Software encounters a procedure error as it tries to execute this subroutine. The procedure error occurs for one of the following reasons:

- You have specified a port that has already been allocated.
- A DECnet server logical link could not be established because of system or DECnet errors.

ARGBLK+2 contains an ASCIZ string pointer to the DECnet object identification (in the format or name or a numeric string) that identifies your process. Your system manager can supply you with the object identification, which is in the X.25 Server Data Base. See the TOPS-10 PSI System Manager's Guide for information on how the X.25 Gateway Software compares data in the X.25 Server Data Base with data in Incoming Call Packets to select which incoming calls should be routed to your process.

PART III
X.29 Reference Guide

CHAPTER 5

THE TOPS-10 X.29 SOFTWARE

The TOPS-10 X.29 Software, X29SRV, connects your terminal - through a PPSN - to a TOPS-10 host. To make this connection, follow the instructions in this chapter, which explain how to:

- Establish a physical connection between your terminal and the PPSN Packet Assembler/Disassembler (PAD) facility.
- Instruct the PAD to create a virtual circuit that connects your terminal to X29SRV on the TOPS-10 host through the X.25 Gateway node.
- Instruct X29SRV to connect your terminal to the TOPS-10 host of your choice.

Figure 5-1 illustrates the major hardware and software components that are needed for these connections.

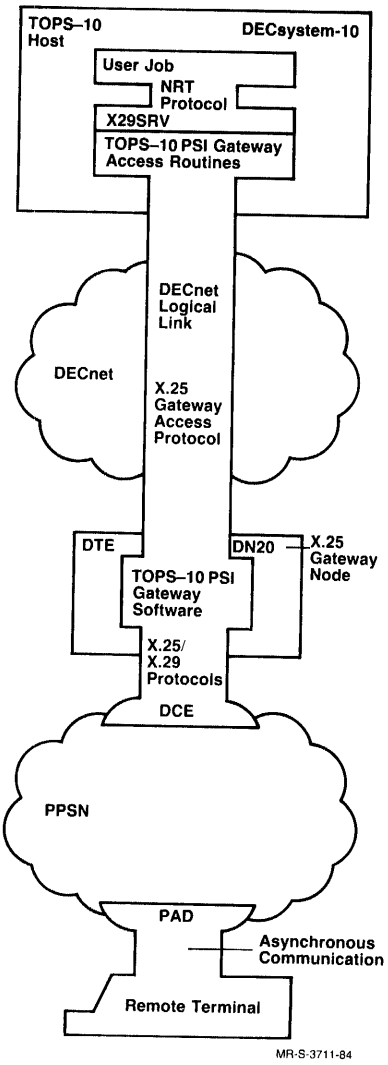


Figure 5-1: X.29 Terminal Access to a TOPS-10 Host

The remote terminal in Figure 5-1 is communicating with a TOPS-10 host that is running both X29SRV and the user job. It is also possible to connect a remote terminal to a TOPS-10 host other than one that is running the X.29 Software. Figure 5-2 illustrates the components needed for such a connection.

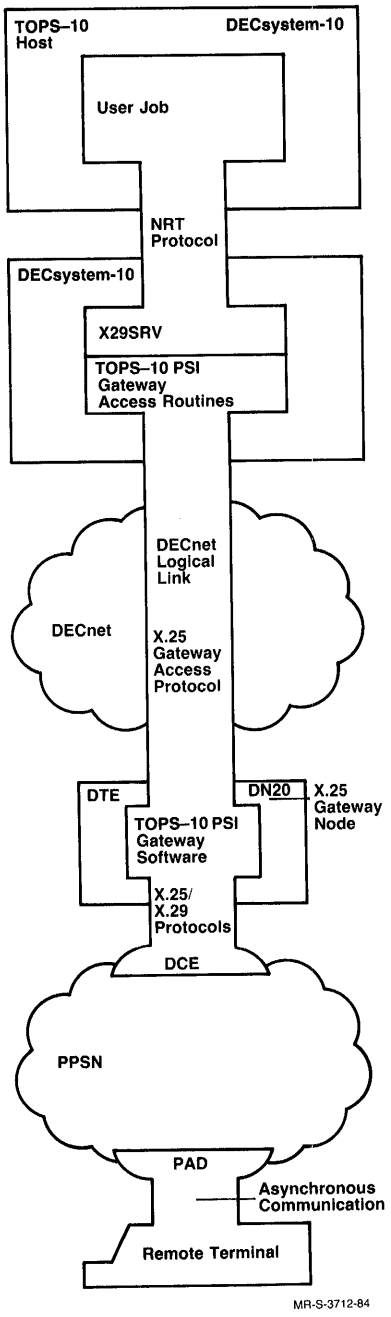


Figure 5-2: X.29 Terminal Access to a TOPS-10 Host that Is Not Running the X.29 Software

5.1 SAMPLE DIALOGUE

To access the TOPS-10 system from a public access PAD on the TELENET network, follow these steps:

1. Dial into TELENET. Your terminal displays a message indicating you have established a connection successfully.

2. Enter D1 as your terminal type at the TERMINAL= prompt.

3. Enter C, which stands for the CONNECT command, and your network address at the @ prompt.

4. Note that X29SRV indicates you have connected to the system successfully.

5. Use the CONNECT command to connect to the DECnet host of your choice, in this case, DRMESQ.

6. Note that your terminal displays the response from the host system.

7. Use the BREAK key to return to X29SRV Command Mode.

8. Use the DISCONNECT command to disconnect from the host.

9. Use the CONNECT command again to connect to another DECnet host, ALBIE.

10. Use the CLEAR command to terminate your session.

The following example shows the terminal session:

```
TELENET      <----------------------------------------------------(1)
617 18I

TERMINAL=D1 (RET)       <-------------------------------------------(2)

@C 61772 (RET)          <-------------------------------------------(3)

617 72 CONNECTED        <-------------------------------------------(4)
Digital Equipment Corporation TOPS-10 PSI Gateway X.29 Server
Friday, April 13, 1984  11:06:31:AM   V1.0(0) #110(00)

X29SRV> CONNECT DRMESQ (RET)       <-------------------------------(5)

RL175D DEC10 Development 11:10:47 TTY52 system 1026/1042      <-----(6)
Connected to Node DRMESQ(26) Line # 52
Please LOGIN or ATTACH

. (BREAK)          <-----------------------------------------------(7)
BREAK
X29SRV> DISC (RET)       <-------------------------------------------(8)

HOST SYSTEM DISCONNECTED
X29SRV> CONNECT ALBIE (RET)          <-----------------------------(9)

CSSE KS 4145 7.02 11:12:29

.
[ Idle line disconnected by host ]

HOST SYSTEM DISCONNECTED
X29SRV> CLEAR (RET)       <-----------------------------------------(10)

DISCONNECTING .....
617 72 DISCONNECTED 00 00 00:00:01:17 30 25
```

5.2 X.3, X.28 AND X.29

The CCITT has endorsed three recommendations that define the form that character terminal access to an X.25 packet switching network should take. These are Recommendations X.3, X.28 and X.29 (described in Section 1.2).

These three recommendations and X.25 together provide a specification for a Packet Assembler/Disassembler (PAD), which interfaces character terminals to an X.25 packet switching network. The CCITT recommendations are:

- Recommendation X.3, which defines the set of parameters that the PAD uses to control your terminal. You can set the parameter values or they can be pre-set in network tables and/or set by the host.

- Recommendation X.28, which defines control procedures used to establish the physical connection to the PPSN, the commands you send to the PAD, and the service signals you receive from the PAD.

- Recommendation X.29, which defines the control messages sent between the PAD and an X29SRV host. All control or PAD messages are special X.25 data packets, called qualified data packets.

- Recommendation X.25, which defines procedures used by the PAD to establish a virtual call to the host, to transmit and receive data packets, and to clear virtual calls.

Figure 5-3 shows how the different protocols work together to support an X.29 terminal connection.

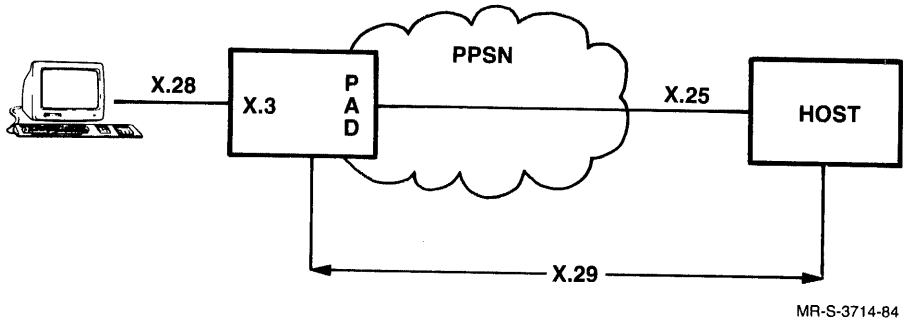


Figure 5-3: X.3, X.25, and X.29 Supporting an X.29 Terminal Connection

5.3 COMMAND LEVELS

You can communicate with a TOPS-10 host system through a PPSN using various software components that belong to either the PPSN or the TOPS-10 PSI Gateway system. For example, assume that your DECsystem-10 is part of the configuration in Figure 5-4:

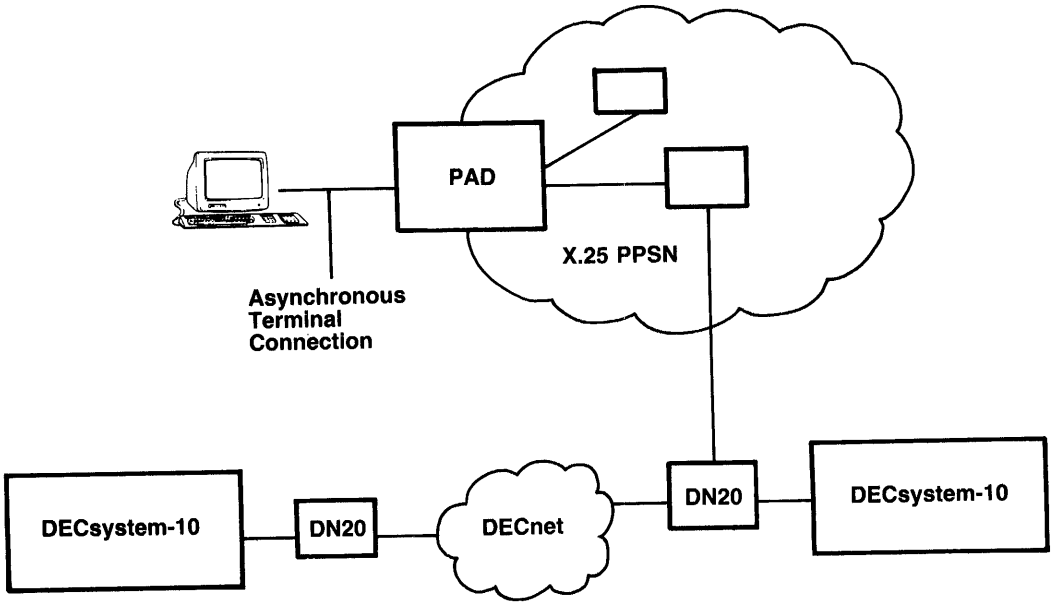


Figure 5-4: TOPS-10 PSI Gateway X.29 Interface

The PAD can be implemented either as a dedicated device with controlling software or as software modules within an existing computer system that has an X.25 interface. Normally, the two software components that you will communicate with prior to logging into your TOPS-10 system are:

- The PAD operating software
- X29SRV residing on one of the TOPS-10 systems in the DECnet network

The PAD operating software is the first level software in the connection. It provides you with a set of commands, one of which lets you request a connection to the TOPS-10 PSI Gateway node on the X.25 PPSN. Each PPSN provides a different set of commands. Refer to your PPSN user's manual for their descriptions.

The TOPS-10 PSI X.29 software is the second level software. It provides you with commands to:

- Request a connection to a TOPS-10 system on the DECnet network
- Disconnect from a connection to a TOPS-10 system on the DECnet network
- Perform other miscellaneous functions

Each software component provides you with two distinct modes of communication:

- Command Mode

  In Command Mode, the software component processes the text you enter at your terminal as user commands. You control the software component in this mode. One of these commands lets you enter the data mode. Another command lets you terminate the connection from the component itself.

- Data Mode

  In Data Mode, most of the text you enter at the terminal is forwarded to the next component as data to be processed by the next component. Usually, a special character or sequence of characters (ESCAPE command) lets you enter the command mode. This may not necessarily be the ESCAPE key at your keyboard. See the appropriate documentation of each component for information about the ESCAPE command in Data Mode.

Thus, while you are communicating with X29SRV in its Command Mode, the PAD software processes your terminal input in its Data Mode. Similarly, while you are connected to the TOPS-10 system, both X29SRV and the PAD software process your terminal input in their respective Data Modes.

Refer to your PPSN user's manual for the descriptions of commands in the Command Mode and the ESCAPE command in the Data Mode.

5.4 COMMUNICATING WITH THE PAD

To connect your terminal to a PAD:

1. Follow instructions in your PPSN user's manual to create a connection between your terminal and the PAD.

2. Follow instructions in your PPSN user's manual to create a connection between the PAD and the TOPS-10 PSI Gateway node, which creates a virtual circuit between your terminal and X29SRV. To follow the instructions in the PPSN user's manual, you will need the DTE address of the X.25 Gateway node; your system manager can supply you with that value.

5.5 COMMUNICATING WITH X29SRV

X29SRV provides you with commands to control your terminal session and the characteristics of the connection.

You can abbreviate all commands. Use the number of characters needed to make the command unique. For example, you can use CL for the CLEAR command or I for the INFORMATION command.

5.5.1 Error Conditions

If you enter a command that is syntactically incorrect, X29SRV responds with the following error message:

```
?Illegal X29SRV command <command text>
 <diagnostic text>
```

where <command text> is the text string that X29SRV acknowledges receiving from you and <diagnostic text> is the reason the command is incorrect. The following are examples of X29SRV detection of syntactical errors. For example, if you misspell the command CONNECT, your terminal displays:

```
X29SRV> CONECT MARKET FDX (RET)

?Illegal X29SRV command "CONECT MARKET FDX"
 Unrecognized switch or keyword: "CONECT"

X29SRV>
```

If you abbreviate a command so that it is not unique, you cause an error condition. For example, there are two commands that begin with the character C -- CLEAR and CONNECT. Therefore, you must shorten the commands to two characters -- CL and CO.

```
X29SRV> C MARKET FDX (RET)

?Illegal X29SRV command "C MARKET FDX"
 Ambiguous switch or keyword: "C"

X29SRV>
```

If you enter a command that is longer than the packet size set up by the network for your terminal connection, the command cannot be processed correctly by X29SRV.

The following sections list the commands in alphabetical order.

5.5.2 CLEAR Command

Description:

Command Mode command.

The CLEAR command disconnects your terminal from X29SRV. You return to the PAD's Command Mode when X29SRV terminates the connection.

Format:

CLEAR

Arguments:

None.

Messages:

DISCONNECTING .....

The PAD software has been requested to disconnect your terminal from X29SRV and put your terminal communication back at the PAD's Command Mode.

Remarks:

None.

Example:

X29SRV> CLEAR (RET)

DISCONNECTING .....

5.5.3 CONNECT Command

Description:

Command Mode command.

The CONNECT command requests a connection to one of the TOPS-10 systems on the DECnet network that is serviced by X29SRV.

Format:

CONNECT host [padset]

Arguments:

host — is the host name of the TOPS-10 system on the DECnet network.

padset — is the name of a set of CCITT and other national X.3 parameters present at the DECsystem-10 running X29SRV. X29SRV uses the values of those parameters to define your terminal characteristics after the connection to the host is established successfully.

Your system manager can supply you with the names of the X.3 parameter sets you can use and the X.3 parameters contained in each set. If you do not include the name of an X.3 parameter set in the command, a default parameter set will be used. For more information on X.3 parameters, see the TOPS-10 PSI System Manager's and Operator's Guide.

Messages:

CONNECTION IS NOT SUPPORTED

X29SRV cannot service the host system that you specified due to incompatible protocols within the DECnet network (usually because that system is not running either the TOPS-10 or TOPS-20 operating system).

FAILED TO CONNECT

The host system that you specified does not exist on the DECnet network or it is temporarily out of service.

UNDEFINED X.3 PARAMETER SET

The name of the X.3 parameter set that you specified in the CONNECT command has not been defined and the default parameter set has been used instead.

TIMED OUT

You took too long to request a connection to a host system.

DISCONNECTING .....

The PAD software has been requested to disconnect your terminal from X29SRV and put your terminal communication back at the PAD's Command Mode.

Remarks:

Depending on how your system manager defines the characteristics of X29SRV, it may allow you to try again after unsuccessfully attempting to connect to a host system. For security reasons, the system manager may choose to have X29SRV terminate a connection after an unsuccessful attempt and you will be back at the PAD's Command Mode.

If you issue the CONNECT command while the connection to a host system is still active, your job on that system will be detached when the new connection is being established.

Example:

```
X29SRV> CONNECT DRMESQ FDX (RET)

RL175D DEC10 Development 17:27:02 TTY54 system 1026/1042
Connected to Node DRMESQ(26) Line   54
Please LOGIN or ATTACH

.
```

5.5.4 DEFINE Command

Description:

Command Mode command.

The DEFINE command lets you change the X.3 parameters that define your terminal connection characteristics.

If you use this command prior to connecting to a TOPS-10 host, the X.3 parameters in the set specified by the CONNECT command (or the default set if none is specified) may supersede the values defined by this command.

Format:

```
DEFINE  DEFAULT
        padset
```

Arguments:

padset	is the name of a set of CCITT and other national X.3 parameters present at the DECsystem-10 running X29SRV. X29SRV uses the values of those parameters to define your terminal characteristics after you resume the terminal connection with the host. Your system manager can supply you with the names of the X.3 parameter sets you can use. He can also tell you which X.3 parameters are contained in each set.

Messages:

UNDEFINED X.3 PARAMETER SET

The parameter set that you specified has not been defined.

None — The command has been executed successfully.

Remarks:

If you specify the DEFAULT keyword, the default parameter set will be used. Your system manager can tell you what X.3 parameters are contained in that set.

Example:

```
X29SRV> DEFINE DEFAULT (RET)
X29SRV> DEFINE FDX (RET)
```

5.5.5 DISCONNECT Command

Description:

Command Mode command.

The DISCONNECT command lets you terminate the connection to the TOPS-10 host. If you have not logged out prior to entering this command, your job at the host system will be detached. You will be back at the X29SRV's Command Mode.

Format:

DISCONNECT

Arguments:

None.

Messages:

HOST SYSTEM DISCONNECTED	The host system disconnected your terminal session successfully.
None	You are not connected to any host system.

Remarks:

None

Example:

X29SRV> DISCONNECT (RET)

HOST SYSTEM DISCONNECTED

5.5.6 INFORMATION Command

Description:

Command Mode command.

The INFORMATION command displays various information related to the terminal connection. Due to security reasons, the system manager may choose to disable this command and not all information will be displayed.

Format:

INFORMATION

Arguments:

None.

Messages:

None

Remarks:

None.

Example:

```
X29SRV> INFORMATION (RET)

9-JUN-84 02:43:29PM  Line 00 At Node 110 (DRMESQ)  -  Connected to DRMESQ
Available X.3 profiles (6):  DEFAULT, EDIT, FDX, HDX, KERMIT, UK
```

5.6 ENTERING COMMAND MODE

When you communicate with the TOPS-10 host system, you are in the Data Mode. You can enter the X29SRV Command Mode by pressing the BREAK key at your keyboard. This triggers the PAD software to transmit a command sequence to X29SRV notifying it of your request. X29SRV responds by temporarily suspending the output from the TOPS-10 host system and starts processing the text you type at your terminal as X29SRV commands (see Section 5.3). If data is passing through the PPSN when you press the BREAK key, it is discarded by the PAD software. A null command (a carriage return without preceding text) lets you resume the Data Mode.

The example below shows the interaction between you and X29SRV during a terminal session. Assume you want to disconnect from your current TOPS-10 host system and connect to another TOPS-10 host system, DRMESQ, on the same DECnet network:

```
.ms (RET)
 24 messages, 27 blocks.
MS>read (message sequence) subject (string) DN20 (RET)
%No messages match this specification
MS>exit (RET)

. (BREAK)

BREAK
X29SRV> CONNECT DRMESQ (RET)

RL175D DEC10 Development 14:27:47 TTY52 system 1055/1056
Connected to Node DRMESQ(26) Line # 52
Please LOGIN or ATTACH

.TTY NO ECHO

.
```

You can resume a connection with a TOPS-10 host system by pressing the RETURN key without any preceding text as follows:

```
X29SRV> CONNECT DRMESQ (RET)

RC702A 7.02 Miso System 09:33:48 TTY44 system 3500
Connected to Node DRMESQ(1) Line # 44
Please LOGIN or ATTACH

.TTY NO ECHO

.logi 07,1055 (RET)

JOB 3 RC702A 7.02 Miso System TTY44
******
09:34     7-Aug-84        Tue

.

.dir/f (BREAK)
BREAK
X29SRV> info (RET)
7-AUG-84 09:34:24AM  Line 00 At Node 169 (DRMESQ), Connected to DRMESQ
Available X.3 profiles (6):  DEFAULT, EDIT, FDX, HDX, KERMIT, UK
X29SRV> (RET)

CONTINUE

  (CTRL/R)  dir/f (RET)

PIT:     [07,1055]
SUBSYS   SFD      PSITST   SFD      X29      SFD      NML      SFD
MAIL     TXT      X25GEN   SFD
DSKO:
DECLAR   CCL      SWITCH   INI

.
```

5.7 TERMINATING A CONNECTION

You can break a connection with your PPSN at any time by hanging up the phone.

APPENDIXES

APPENDIX A

SAMPLE FORTRAN-10 PROGRAMS

This appendix contains sample FORTRAN-10 programs that communicate with each other through a PPSN. The programs are:

- SVCRCV, which accepts a Switched Virtual Circuit call and receives data sent through the circuit.
- SVCXMI, which initiates a Switched Virtual Circuit call and sends data through the circuit.

Section A.1 explains features of the two programs, while Section A.2 contains a listing of SVCRCV, and Section A.3 contains a listing of SVCXMI.

A.1 PROGRAM FEATURES

This section explains and gives hints related to individual features of each program.

A.1.1 Receiving Program

The program SVCRCV is the passive partner (slave) of the transmitting program, SVCXMI. SVCRCV starts by declaring itself available to receive an incoming call from the X.25 gateway node. The program does this by calling the subroutine X25WIC (Wait Incoming Call):

```
PROGRAM SVCRCV
EXTERNAL X25WIC
INTEGER CSTATE, NSTATE, PORT, RESULT
DIMENSION WORKSP(172)
CSTATE = 0
CALL X25WIC ('EXAMPLE', WORKSP, PORT, RESULT)
IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
CSTATE = 3
NSTATE = ISTAT (PORT, CSTATE)
IF (NSTATE .NE. 4) CALL ABORT (PORT, RESULT, NSTATE)
```

SVCRCV initializes the current port state, CSTATE, as 0 (UNDEFINED) before calling the subroutine X25WIC. After the subroutine returns successfully (return code 0), the program sets the current port state to 3 (LISTENING). Note that there are other subroutines that change the port state upon successful execution. When one of these subroutines has been successfully executed, you do not need to execute X25RPS to determine the port state. For example:

- X25RCV (RESET VIRTUAL CIRCUIT), which changes the port state from UNSYNC to RUNNING.
- X25NCS (NO COMMUNICATION SEEN), which changes the port state from NO COMMUNICATION to RUNNING.

The chosen length of the data packets in this example is 128 bytes. The length of the array WORKSP (which is to be used by the TOPS-10 PSI Gateway Access routines as working space) is calculated as follows:

```
140 + (128/4) = 172
```

The subroutine X25WIC declares the user task to be a DECnet target task that is available for network dialogue with the PSI Gateway Software and uses the object name EXAMPLE. After the subroutine returns successfully, NSP assigns a logical link to the user task. Contact your system manager for the appropriate object names and/or object numbers on your system.

The program defines ABORT as a subroutine that displays the current port state and error, terminates port access, and stops the program:

```
      SUBROUTINE ABORT (PORT, CODE, STATE)
      EXTERNAL X25TPA
      INTEGER PORT, CODE, STATE
      WRITE (5,1000) CODE, STATE
1000  FORMAT (/,' * ERROR #',I2,', current port state ',I2,' *',/)
      CALL X25TPA (PORT, 0)
      STOP
      END
```

After the X.25 port is allocated to wait for an incoming call, the program calls the integer function ISTAT to poll for the port state until it has changes:

```
      INTEGER FUNCTION ISTAT (PORT, STATE)
      EXTERNAL X25RPS, IDLE
      INTEGER PORT, STATE, PSTATE, RESULT
100   CALL X25RPS (PORT, 0, PSTATE, 0, 0, 0, 0, 0, RESULT)
      IF (RESULT .EQ. 0) GOTO 200
      ISTAT = 10
      RETURN
200   IF (PSTATE .NE. STATE) GOTO 300
      CALL IDLE (5)
      GOTO 100
300   ISTAT = PSTATE
      RETURN
      END
```

At times, you will find it necessary to wait for the circuit to change from one state to another (for example, from LISTENING to CALLED) before any processing can be done for that circuit. While waiting, it is advisable to incorporate a MACRO-10 routine into your code that puts your program into a dormant state before calling subroutine X25RPS to check the status of the circuit. Constant calling of subroutine X25RPS to check the status of the circuit tends to create unnecessary CPU activity.

The function ISTAT calls the subroutine IDLE to perform that task:

```
T1=1
T2=2
AP=16
SP=17

            SEARCH  UUOSYM

IDLE::      PUSH    SP,T1           ;Save temporary registers
            PUSH    SP,T2
            MOVE    T2,0(AP)        ;Get FORTRAN parameter entry
            MOVE    T2,0(T2)        ;Get the idle interval in seconds
            MOVSI   T1,(HB.SEC)
            ADD     T1,T2
            HIBER   T1,             ;Sleep
            JFCL
            POP     SP,T2           ;Restore temporary registers
            POP     SP,T1
            POPJ    SP,0            ;Return
            END
```

After the incoming call is received, the program accepts it by calling the subroutine X25AIC (Accept Incoming Call). In this instance, you want to accept the incoming call unconditionally; you can, therefore, ignore the content of the incoming call packet and proceed to call subroutine X25AIC. However, you can check the contents of the incoming call packet by calling the subroutine X25RIC (Read Incoming Call) before establishing the circuit. The program accepts the incoming call without specifying the user group, accept data, and facilities. Those null parameters are represented by 0:

```
EXTERNAL X25AIC
CSTATE = 4
CALL X25AIC (PORT, 0, 0, 0, 0, 0, RESULT)
IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
CSTATE = 5
```

After the subroutine X25AIC returns successfully, the programmer correctly assume that the port state has changes from 4 (CALLED) to 5 (RUNNING). The program is ready to transfer data.

It has been agreed that after the circuit is established, the program SVCXMI will send an interrupt byte of any value to indicate that it is ready to transmit data over the circuit. The program SVCRCV attempts to read that interrupt byte:

```
      .
      .
      EXTERNAL IDLE, X25RPS
      INTEGER IBYTE
      LOGICAL IAVAIL
      .
      .
100   CALL X25RPS (PORT, 0, 0, 0, 0, 0, IAVAIL, 0, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      IF (IAVAIL .EQ. .TRUE.) GOTO 110
      CALL IDLE (1)
      GOTO 100
110   CALL X25RIM (PORT, IBYTE, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      WRITE (5,1100) IBYTE
1100  FORMAT (/,' Received interrupt byte "',O3,/)
      .
      .
```

The private protocol, which has been agreed between the two programs, calls for the program SVCRCV not to confirm the interrupt byte that it has just received until it receives all of the data sent by its partner, SVCXMI.

The main segment of the program SVCRCV is the subroutine RCVCON, which reads and interprets the data that the program receives from the network. RCVCON interprets data according to the following conventions:

- A data record is a set of data packets of which the last data packet has the more bit not set.
- Each data record is preceded by a "control" data packet containing one 8-bit byte that indicates the conversion mode. This user-defined conversion mode is in the range 1 to 9 and indicates to the receiving program how it should read and interpret the following data record.
- If the "control" data packet contains a control-Z character (octal 32), the transmission is complete.

The subroutine RCVCON declares the dimension of the receiving data array to be 512 words, which is long enough to ensure that there is enough room for all types of data. You should supply an array buffer large enough to accommodate at least one packet of data with any data type conversion mode. You can obtain the current packet size by calling the subroutine X25RPS.

The resulting length of the receiving array can vary from one data type conversion mode to another. For example, assume you have full packet of 128 data bytes with the more bit not set is pending to be read. If the subroutine X25RDM (READ DATA MESSAGE) is called with data type 0 (full 36 bits), it will return a data array of 29

elements. However, if the subroutine is called with data type 5 (leftmost 7 bits) it will return an array of 128 elements. This read conversion of the data, which is done by the subroutine X25RDM, is symmetrical to the write conversion procedure, which is done by the subroutine X25SDM:

```
      SUBROUTINE RCVCON (PORT)
      EXTERNAL X25RDM
      INTEGER PORT, QBIT, CNVRSN, RESULT, LENGTH, REMLEN
      LOGICAL MBIT
      INTEGER BUFFER(512)
      INTEGER IBUF(512)
      REAL XBUF(512)
      EQUIVALENCE (XBUF(1), BUFFER(1)), (IBUF(1), BUFFER(1))
100   CALL WTDATA (PORT)
      LENGTH = 512
      CALL X25RDM (PORT, 4, BUFFER, LENGTH, QBIT, MBIT, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, 5)
      IF (LENGTH .LE. 0) GOTO 100
      IF (BUFFER(1) .EQ. "32) RETURN
      CNVRSN = BUFFER(1)
      REMLEN = 512
150   CALL WTDATA (PORT)
      I = 512 - REMLEN + 1
      LENGTH = REMLEN
      CALL X25RDM (PORT, CNVRSN, BUFFER(I), LENGTH, QBIT, MBIT, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, 5)
      IF (LENGTH .LE. 0) GOTO 100
      GOTO (200,200,200,200,200,500,500,500,500,500) (CNVRSN+1)
200   REMLEN = REMLEN - LENGTH
      IF (MBIT .EQ. .TRUE.) GOTO 150
      LENGTH = 512 - REMLEN
      GOTO (210,220,220,220,220) (CNVRSN + 1)
210   DO 212 I = 1,LENGTH,2
      WRITE (5,2100) XBUF(I), XBUF(I+1)
2100  FORMAT (' ',F10.6,' ',F10.6)
212   CONTINUE
      GOTO 100
220   DO 222 I = 1,LENGTH
      WRITE (5,2200) IBUF(I)
2200  FORMAT (I10)
222   CONTINUE
      GOTO 100
500   ITEMP1 = CNVRSN - 5 + 1
      ITEMP2 = MOD (LENGTH, ITEMP1)
      REMLEN = REMLEN - ((LENGTH / ITEMP1) + ITEMP2)
      IF (MBIT .EQ. .TRUE.) GOTO 150
      LENGTH = 512 - REMLEN
      GOTO (510,520,530,540,550) ITEMP1
510   WRITE (5,5100) (IBUF(I),I=1,LENGTH)
5100  FORMAT (1H ,512A1,$)
      GOTO 100
520   WRITE (5,5200) (IBUF(I),I=1,LENGTH)
5200  FORMAT (1H ,512A2,$)
      GOTO 100
530   WRITE (5,5300) (IBUF(I),I=1,LENGTH)
5300  FORMAT (1H ,512A3,$)
      GOTO 100
540   WRITE (5,5400) (IBUF(I),I=1,LENGTH)
5400  FORMAT (1H ,512A4,$)
      GOTO 100
550   WRITE (5,5500) (IBUF(I),I=1,LENGTH)
5500  FORMAT (1H ,512A5,$)
      GOTO 100
      END
```

Before each call of the subroutine X25RDM, the program calls the subroutine WTDATA to poll for incoming data from the network. This subroutine is similar to the function ISTAT, which checks for the port state transition:

```
      SUBROUTINE WTDATA (PORT)
      EXTERNAL IDLE, X25RPS
      INTEGER PORT, RESULT
      LOGICAL DAVAIL
100   CALL X25RPS (PORT, 0, 0, 0, 0, DAVAIL, 0, 0, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, 5)
      IF (DAVAIL .EQ. .TRUE.) RETURN
      CALL IDLE (1)
      GOTO 100
      END
```

After the subroutine RCVCON is executed successfully, the main program calls the subroutine X25CIM (CONFIRM INTERRUPT MESSAGE) to confirm the interrupt received at the beginning of the connection:

```
      CALL X25CIM (PORT, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
```

At this point, the program SVCRCV has performed all of its functions and is ready to terminate the circuit. However, before doing so, this program, the passive partner of the pair, must wait for the circuit to be cleared by the other program:

```
      .
      .
      EXTERNAL X25RCD, X25TPA
      INTEGER CAUSE, DIAGNO
      .
      .
      NSTATE = ISTAT (PORT, CSTATE)
      IF (NSTATE .NE. 9) GOTO 200
      CALL X25RCD (PORT, CAUSE, DIAGNO, 0, 0, 0, 0, RESULT)
      WRITE (5,1110) CAUSE, DIAGNO
1110  FORMAT (' Port is cleared, cause "',O3,', diagnostic "',O3)
200   CALL X25TPA (PORT, RESULT)
      STOP
      END
```

A.1.2 Transmitting Program

The program SVCXMI is the active partner (master) of the receiving program, SVCRCV. The program initiates all operations; that is, it initiates the Switched Virtual Circuit call, the data transfer, and the clearing of the virtual circuit after receiving the confirmation from its partner that the data has been received.

The program communicates with the user to acquire text data and generates "number crunching type" data to transfer to the receiving program. This program covers all data types, both text and binary (integer and real), which can be transferred over an X.25 circuit with the FORTRAN-10 PSI Gateway Access routines.

The program begins by prompting the user for the remote DTE address, which the program uses in placing the virtual circuit call:

```
      PROGRAM SVCXMI
      .
      .
      INTEGER LENGTH, DTE(4)
      .
      .
100   WRITE (5,1000)
1000  FORMAT (' DTE address: ',$)
      READ (5,1010) LENGTH, (DTE(I),I=1,4)
1010  FORMAT (Q,4A5)
      IF (LENGTH .LE. 0) GOTO 100
      .
      .
```

The program SVCXMI calls the subroutine X25ISC (Initiate Switched Circuit) to establish the virtual circuit:

```
      .
      .
      EXTERNAL X25ISC
      DIMENSION WORKSP(172)
      INTEGER CSTATE, NSTATE, PORT
      .
      .
      CSTATE = 0
      CALL X25ISC (8HTELENET , 'X25-GATE ', DTE,
     $             0, 0, 0, 0, 0, 0, WORKSP, PORT, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      CSTATE = 2
      .
      .
```

After the subroutine X25ISC returns successfully, the programmer assumes that the port state has changes from 0 (UNDEFINED) to 2 (CALLING) without calling the subroutine X25RPS to obtain the new port state.

In the above call of the subroutine X25ISC, the programmer chose not to supply the calling DTE address, user group name, user data, or facilities. Those are specified as null parameters (0s). An ASCII string parameter may be passed to the X.25 Gateway Access subroutines as a variable name, Hollerith constant, or character constant. Since FORTRAN-10 does not allow null parameters in the argument list of a CALL statement, specify a null string with the value 0.

After the call request packet is sent to the network, the program calls the integer function ISTAT, which is described in Section A.1.1, and polls for the port state until it has changes:

```
      .
      .
      NSTATE = ISTAT (PORT, CSTATE)
      IF (NSTATE .NE. 5) CALL ABORT (PORT, 0, NSTATE)
      .
      .
```

The program SVCXMI now sends an interrupt byte to its partner to indicate that it is ready to start transferring data. The program prompts you for the value of the interrupt byte:

```
      .
      .
      INTEGER IBYTE
      .
      .
200   CSTATE = 5
      WRITE (5,2000)
2000  FORMAT (' Interrupt byte (octal): ',$)
      READ (5,2010) IBYTE
2010  FORMAT (O3)
      CALL X25SIM (PORT, IBYTE, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      .
      .
```

This program demonstrates different ways of sending ASCII data, such as text entered by the user and read in A1 format and an ASCII file read in A5 format.

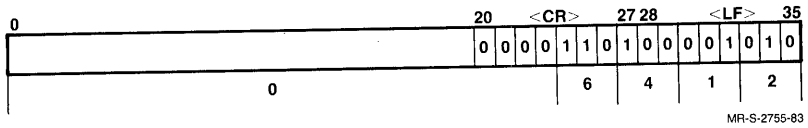
Before each record of text is transmitted over the circuit, SVCXMI notifies its partner of the type of transmitted data. The program does this by sending a "control" data packet containing one byte that specifies the data type (see Section A.1.1 for a description of the private protocol between the two programs). After you enter the text, the program stores each ASCII character in the leftmost 7 bits of each array element of the array BUFFER. The variable LENGTH contains the number of characters you enter:

```
      .
      .
      INTEGER BUFFER(256)
      .
      .
      CALL X25SDM (PORT, 4, 5, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
300   WRITE (5,3000)
3000  FORMAT (' Text string: ',$)
      READ (5,3010,ERR=300) LENGTH, (BUFFER(I),I=1,256)
3010  FORMAT (Q,256A1)
      IF (LENGTH .LE. 0) GOTO 300
      CALL X25SDM (PORT, 5, BUFFER, LENGTH, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      .
      .
```

The program now tries to send a portion of an ASCII file over the circuit:

```
      .
      .
      INTEGER FILBUF(2)
      DOUBLE PRECISION FILNAM
      EQUIVALENCE (FILBUF(1), FILNAM)
      .
      .
400   WRITE (5,4000)
4000  FORMAT (' File name: ',$)
      READ (5,4010) (FILBUF(I),I=1,2)
4010  FORMAT (2A5)
      OPEN (UNIT=1,DEVICE='DSK',FILE=FILNAM,ACCESS='SEQIN',ERR=440)
      CALL X25SDM (PORT, 4, 9, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) GOTO 440
      DO 420 M = 1,20
      READ (1,4020,END=430,ERR=430) LENGTH, (BUFFER(I),I=1,256)
4020  FORMAT (Q,256A5)
      IF (LENGTH .LE. 0) GOTO 410
      CALL X25SDM (PORT, 9, BUFFER, LENGTH, 0, .TRUE., RESULT)
      IF (RESULT .NE. 0) GOTO 440
410   CALL X25SDM (PORT, 3, "6412, 1, 0, .TRUE., RESULT)
      IF (RESULT .NE. 0) GOTO 440
420   CONTINUE
430   CALL X25SDM (PORT, 9, BUFFER, 0, 0, .FALSE., RESULT)
440   CLOSE (UNIT=1)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      .
      .
```

After each text line from the ASCII file, the program sends a pair of carriage return and line feed characters in the rightmost 16 bits format (statement 410). The characters are passed to the subroutine in the following format:



The rightmost 16 bits of the above 36-bit word (6412 octal) are formatted into 2 octets of the data packet (bits 20-27 form the first octet and bits 28-35 form the second). When those two octets are received and interpreted by the receiving program, they are converted into 7-bit ASCII characters that become carriage return (15 octal) and line feed (12 octal) characters to be displayed.

The above example demonstrates how to manipulate and transmit the data with mixed data conversion modes. Even though the read and write conversion procedures are symmetrical, you can read the data with a data type that differs from the one used in sending the data for some conversion formats. For example, you can use any of the ASCII data formats (5 to 9) and some others (e.g. rightmost 16 bits as above, with care) interchangeably to send and read the text data.

However, take extra care with other binary data conversion modes. The following examples demonstrate ways of sending various types of binary data. For those types of data, it is advisable to use the same conversion mode to send and receive the data.

In the following example, the program generates its own data to be sent to the receiving program. The sending program calculates the first 25 Fibonacci numbers and sends 10 copies of that set of numbers with data type 3 (rightmost 16 bits).

The Fibonacci sequence starts:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Each new term is obtained by taking the sum of the two previous terms. If the first term of the sequence is F(0), then

F(0) = 0, F(1) = 1

and in general

F(n) = F(n-1) + F(n-2), (n .GE. 2)

The largest Fibonacci number in this set is 46368 (132440 octal), which occupies a maximum of 16 bits of the array element. The FORTRAN data type of the numbers is integer.

```
      .
      .
      INTEGER IBUF(256)
      EQUIVALENCE (IBUF(1), BUFFER(1))
      .
      .
      IBUF(1) = 0
      IBUF(2) = 1
      DO 500 I = 3,25
      IBUF(I) = IBUF(I-1) + IBUF(I-2)
500   CONTINUE
      .
      .
```

First, the sending program sends an identification text string to the receiving program so that it can notify you of the nature of the received data. Notice that the length of the text string, 30, is the number of characters in the string, not the number of 36-bit words the string comprises:

```
      .
      .
      CALL X25SDM (PORT, 4, 9, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      CALL X25SDM (PORT, 9, 'The first 25 Fibonacci numbers', 30,
     $             0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      .
      .
```

The program sends 10 copies of the Fibonacci numbers as one data record:

```
      .
      .
      CALL X25SDM (PORT, 4, 3, 1, 0, .FALSE., RESULT)
      DO 510 I = 1,10
      CALL X25SDM (PORT, 3, IBUF, 25, 0, .TRUE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
510   CONTINUE
      CALL X25SDM (PORT, 3, IBUF, 0, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      .
      .
```

The data type 0 (36 bits) is ideal for transferring arrays of binary data (REAL, DOUBLE PRECISION, etc.). The following example sends an array of the first 128 numbers and their square roots. The program generates the values of the array:

```
      .
      .
      REAL NUMBER, XBUF(256)
      EQUIVALENCE (XBUF(1), BUFFER(1))
      .
      .
      NUMBER = 1.0
      DO 600 I = 1,256,2
      XBUF(I) = NUMBER
      XBUF(I+1) = SQRT (NUMBER)
      NUMBER = NUMBER + 1.0
600   CONTINUE
      .
      .
```

The program sends the entire array by calling the subroutine X25SDM only once:

```
      .
      .
      CALL X25SDM (PORT, 4, 9, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      CALL X25SDM (PORT, 9, 30HNumbers and their Square Roots, 30,
     $             0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      CALL X25SDM (PORT, 4, 0, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      CALL X25SDM (PORT, 0, XBUF, 256, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      .
      .
```

The program concludes the data transfer session by sending a control-Z character (32 octal) to its partner, then waits for the confirmation of the interrupt byte it sent at the beginning of the session:

```
      CALL X25SDM (PORT, 4, "32, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      CALL WTCFRM (PORT, 300)
```

The subroutine WTCFRM is defined:

```
      SUBROUTINE WTCFRM (PORT, WAIT)
      EXTERNAL X25RPS, IDLE
      INTEGER PORT, WAIT, RESULT
      LOGICAL PENDNG
      DO 100 I = 1,WAIT
      CALL X25RPS (PORT, 0, 0, 0, PENDNG, 0, 0, 0, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, 5)
      IF (PENDNG .EQ. .FALSE.) RETURN
      CALL IDLE (1)
100   CONTINUE
      RETURN
      END
```

At this point, the receiving program acknowledges the end of transmission. The sending program clears the virtual circuit and terminates:

```
      CALL X25CSC (PORT, "252, 0, 0, 0, 0, 0, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      CSTATE = 8
      CSTATE = ISTAT (PORT, CSTATE)
      CALL X25TPA (PORT, RESULT)
      STOP
      END
```

The sending program clears the virtual circuit with the user diagnostic octal 252. The program then waits for the network to confirm the clear request by calling the function ISTAT, which is described in Section A.1.1.

A.2 LISTING OF SAMPLE PROGRAM SVCRCV.FOR

```
      PROGRAM SVCRCV

C Storage declarations
C
      EXTERNAL IDLE, X25AIC, X25RCD, X25RPS, X25TPA, X25WIC
      INTEGER CSTATE, NSTATE, PORT, RESULT, IBYTE, CAUSE, DIAGNO
      LOGICAL IAVAIL
      DIMENSION WORKSP(172)

C Declare self to be available to receive an incoming call
C
      CSTATE = 0
      CALL X25WIC ('EXAMPLE', WORKSP, PORT, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Check port state and wait for port state to become CALLED, then
C proceed and accept the incoming call
C
      CSTATE = 3
      NSTATE = ISTAT (PORT, CSTATE)
      IF (NSTATE .NE. 4) CALL ABORT (PORT, RESULT, NSTATE)

C Accept incoming call unconditionally
C
      CSTATE = 4
      CALL X25AIC (PORT, 0, 0, 0, 0, 0, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      CSTATE = 5

C Check only interrupt available flag, ignore other indicators
C
100   CALL X25RPS (PORT, 0, 0, 0, 0, 0, IAVAIL, 0, RESULT)

C Check if we have detected the interrupt byte
C
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      IF (IAVAIL .EQ. .TRUE.) GOTO 110

C Idle process for 1 second before checking the circuit status again
C
      CALL IDLE (1)
      GOTO 100

C Read interrupt message
C
110   CALL X25RIM (PORT, IBYTE, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Display the received interrupt byte in octal
C
      WRITE (5,1100) IBYTE
1100  FORMAT (/,' Received interrupt byte "',O3,/)

C Read data and perform conversion for display
C
      CALL RCVCON (PORT)

C Confirm interrupt that we received to confirm end of data reception
C
      CALL X25CIM (PORT, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Wait until port is cleared. If port state becomes anything else,
```

```
C go ahead and terminate port access anyway.
C
      NSTATE = ISTAT (PORT, CSTATE)
      IF (NSTATE .NE. 9) GOTO 200

C Read clear cause and diagnostic, ignore the data and facilities
C
      CALL X25RCD (PORT, CAUSE, DIAGNO, 0, 0, 0, 0, RESULT)
      WRITE (5,1110) CAUSE, DIAGNO
1110  FORMAT (' Port is cleared, cause "',O3,', diagnostic "',O3)

C Terminate port access and program
C
200   CALL X25TPA (PORT, RESULT)

      STOP
      END

      SUBROUTINE RCVCON (PORT)

C+
C DESCRIPTION   Read and display received data.
C
C PARAMETERS    PORT    Port number
C-

      EXTERNAL X25RDM

      INTEGER PORT, QBIT, CNVRSN, RESULT, LENGTH, REMLEN
      LOGICAL MBIT

      INTEGER BUFFER(512)
      INTEGER IBUF(512)
      REAL XBUF(512)
      EQUIVALENCE (XBUF(1), BUFFER(1)), (IBUF(1), BUFFER(1))

C Wait for protocol data byte
C
100   CALL WTDATA (PORT)

C Set length of the receiving buffer and read protocol byte
C
      LENGTH = 512
      CALL X25RDM (PORT, 4, BUFFER, LENGTH, QBIT, MBIT, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, 5)

C Ignore null string
C
      IF (LENGTH .LE. 0) GOTO 100

C Check if this is the end of data reception
C
      IF (BUFFER(1) .EQ. "32) RETURN

C Record the data conversion code
C
      CNVRSN = BUFFER(1)

C Initialize buffer length
C
      REMLEN = 512

C Wait for actual data buffer
```

```
C
150   CALL WTDATA (PORT)

C Calculate the starting address of the receiving buffer
C
      I = 512 - REMLEN + 1
      LENGTH = REMLEN

C Read data record
C
      CALL X25RDM (PORT, CNVRSN, BUFFER(I), LENGTH, QBIT, MBIT, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, 5)
      IF (LENGTH .LE. 0) GOTO 100

C Dispatch processing code
C
      GOTO (200,200,200,200,200,500,500,500,500,500) (CNVRSN+1)

C Calculate the remaining length of the receiving buffer
C
200   REMLEN = REMLEN - LENGTH

C If there are more data to read, go back and continue reading using
C the same conversion format, otherwise display received data at the
C controlling terminal
C
      IF (MBIT .EQ. .TRUE.) GOTO 150
      LENGTH = 512 - REMLEN
      GOTO (210,220,220,220,220) (CNVRSN + 1)

C Binary data: 36 bits
C
210   DO 212 I = 1,LENGTH,2
      WRITE (5,2100) XBUF(I), XBUF(I+1)
2100  FORMAT (' ',F10.6,' ',F10.6)
212   CONTINUE
      GOTO 100

C Binary data: high 32 bits, low 32 bits, low 16 bits and low 8 bits
C
220   DO 222 I = 1,LENGTH
      WRITE (5,2200) IBUF(I)
2200  FORMAT (I10)
222   CONTINUE
      GOTO 100

C Calculate the remaining length of the receiving buffer
C
500   ITEMP1 = CNVRSN - 5 + 1
      ITEMP2 = MOD (LENGTH, ITEMP1)
      REMLEN = REMLEN - ((LENGTH / ITEMP1) + ITEMP2)

C If there are more data to read, go back and continue reading using
C the same conversion format, otherwise display received data at the
C controlling terminal
C
      IF (MBIT .EQ. .TRUE.) GOTO 150
      LENGTH = 512 - REMLEN
      GOTO (510,520,530,540,550) ITEMP1

C ASCII data in A1 format
```

```
C
510   WRITE (5,5100) (IBUF(I),I=1,LENGTH)
5100  FORMAT (1H ,512A1,$)
      GOTO 100

C ASCII data in A2 format
C
520   WRITE (5,5200) (IBUF(I),I=1,LENGTH)
5200  FORMAT (1H ,512A2,$)
      GOTO 100

C ASCII data in A3 format
C
530   WRITE (5,5300) (IBUF(I),I=1,LENGTH)
5300  FORMAT (1H ,512A3,$)
      GOTO 100

C ASCII data in A4 format
C
540   WRITE (5,5400) (IBUF(I),I=1,LENGTH)
5400  FORMAT (1H ,512A4,$)
      GOTO 100

C ASCII data in A5 format
C
550   WRITE (5,5500) (IBUF(I),I=1,LENGTH)
5500  FORMAT (1H ,512A5,$)
      GOTO 100
      END

      SUBROUTINE WTDATA (PORT)

C+
C DESCRIPTION   Read port status and wait for incoming data indication
C
C PARAMETERS    PORT    Port number
C-

      EXTERNAL IDLE, X25RPS

      INTEGER PORT, RESULT
      LOGICAL DAVAIL

C Check only data available flag, ignore other indicators
C
100   CALL X25RPS (PORT, 0, 0, 0, 0, DAVAIL, 0, 0, RESULT)

C Check if we have detected the incoming data
C
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, 5)
      IF (DAVAIL .EQ. .TRUE.) RETURN

C Idle process for 1 second before checking the circuit status again
C
      CALL IDLE (1)
      GOTO 100
      END

      SUBROUTINE ABORT (PORT, CODE, STATE)

C+
C DESCRIPTION        Terminate communication and abort the process.
```

```
C
C PARAMETERS PORT     Port number.
C                     CODE     Error code.
C                     STATE    Current port state.
C
C-

      EXTERNAL X25TPA
      INTEGER PORT, CODE, STATE

      WRITE (5,1000) CODE, STATE
1000  FORMAT (/,' * ERROR #',I2,', current port state ',I2,' *',/)

      CALL X25TPA (PORT, 0)

      STOP
      END

      INTEGER FUNCTION ISTAT (PORT, STATE)

C+
C DESCRIPTION    Read port status.
C
C PARAMETERS     PORT     Port number.
C                STATE    Current port state.
C
C RETURN         New port state when it is changes; or ERROR port state
C                if failed to read port status.
C-

      EXTERNAL X25RPS, IDLE
      INTEGER PORT, STATE, PSTATE, RESULT

C Check only port state, ignore other indicators
C
100   CALL X25RPS (PORT, 0, PSTATE, 0, 0, 0, 0, 0, RESULT)

C If failed to read port status, return port state ERROR
C
      IF (RESULT .EQ. 0) GOTO 200
      ISTAT = 10
      RETURN

C If the port state has changes, return the new port state
C
200   IF (PSTATE .NE. STATE) GOTO 300

C Otherwise, idle the process for 5 seconds before checking
C the port state again
C
      CALL IDLE (5)
      GOTO 100

C Return new port state
C
300   ISTAT = PSTATE
      RETURN
      END
```

A.3 LISTING OF SAMPLE PROGRAM SVCXMI.FOR

```
      PROGRAM SVCXMI

C Storage declarations
C
      EXTERNAL IDLE, X25CSC, X25ISC, X25SDM, X25SIM, X25TPA
      INTEGER CSTATE, NSTATE, PORT, RESULT, IBYTE, LENGTH
      INTEGER DTE(4), BUFFER(256), IBUF(256), FILBUF(2)
      REAL NUMBER, XBUF(256)
      DOUBLE PRECISION FILNAM
      EQUIVALENCE (FILBUF(1), FILNAM)
      EQUIVALENCE (IBUF(1), BUFFER(1)), (XBUF(1), BUFFER(1))
      DIMENSION WORKSP(172)

C Prompt for remote DTE address
C
100   WRITE (5,1000)
1000  FORMAT (' DTE address: ',$)
      READ (5,1010) LENGTH, (DTE(I),I=1,4)
1010  FORMAT (Q,4A5)
      IF (LENGTH .LE. 0) GOTO 100

C Initiate the virtual circuit to the network
C
      CSTATE = 0
      CALL X25ISC (8HTELENET , 'X25-GATE ', DTE,
     $             0, 0, 0, 0, 0, 0, WORKSP, PORT, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      CSTATE = 2

C Check port state and wait for port state to become RUNNING
C
      NSTATE = ISTAT (PORT, CSTATE)
      IF (NSTATE .NE. 5) CALL ABORT (PORT, 0, NSTATE)

C Send an interrupt to indicate start of transmission. The remote end
C will confirm this interrupt after it receives all data successfully
C
200   CSTATE = 5
      WRITE (5,2000)
2000  FORMAT (' Interrupt byte (octal): ',$)
      READ (5,2010) IBYTE
2010  FORMAT (O3)
      CALL X25SIM (PORT, IBYTE, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Send protocol byte to SVCRCV to indicate data conversion mode
C
      CALL X25SDM (PORT, 4, 5, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Read input data from terminal using A1 format descriptor
```

```
C
300   WRITE (5,3000)
3000  FORMAT (' Text string: ',$)
      READ (5,3010,ERR=300) LENGTH, (BUFFER(I),I=1,256)
3010  FORMAT (Q,256A1)

C Do not attempt to send null string
C
      IF (LENGTH .LE. 0) GOTO 300

C Send the text string in A1 format to SVCRCV.
C
      CALL X25SDM (PORT, 5, BUFFER, LENGTH, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Read file name
C
400   WRITE (5,4000)
4000  FORMAT (' File name: ',$)
      READ (5,4010) (FILBUF(I),I=1,2)
4010  FORMAT (2A5)

C Open input file
C
      OPEN (UNIT=1,DEVICE='DSK',FILE=FILNAM,ACCESS='SEQIN',ERR=440)

C Send protocol byte to SVCRCV to indicate data conversion mode
C
      CALL X25SDM (PORT, 4, 9, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) GOTO 440

C Send the first 20 lines of the file to SVCRCV.
C
      DO 420 M = 1,20
      READ (1,4020,END=430,ERR=430) LENGTH, (BUFFER(I),I=1,256)
4020  FORMAT (Q,256A5)
      IF (LENGTH .LE. 0) GOTO 410
      CALL X25SDM (PORT, 9, BUFFER, LENGTH, 0, .TRUE., RESULT)
      IF (RESULT .NE. 0) GOTO 440

C Follow each line with a pair of <CR><LF> in the rightmost 16 bits
C
410   CALL X25SDM (PORT, 3, "6412, 1, 0, .TRUE., RESULT)
      IF (RESULT .NE. 0) GOTO 440
420   CONTINUE

C Flush the buffered data to SVCRCV by calling X25SDM with 0 length
C
430   CALL X25SDM (PORT, 9, BUFFER, 0, 0, .FALSE., RESULT)

C Close input file
C
440   CLOSE (UNIT=1)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Initialize first 2 Fibonacci numbers
C
      IBUF(1) = 0
      IBUF(2) = 1

C Calculate the next 23 Fibonacci numbers
```

```
C
      DO 500 I = 3,25
      IBUF(I) = IBUF(I-1) + IBUF(I-2)
500   CONTINUE

C Send the sequence to the receiving program
C
      CALL X25SDM (PORT, 4, 9, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

      CALL X25SDM (PORT, 9, 'The first 25 Fibonacci numbers', 30,
     $             0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Send protocol byte to SVCRCV to indicate data conversion mode
C
      CALL X25SDM (PORT, 4, 3, 1, 0, .FALSE., RESULT)

C Send 10 copies of the set of Fibonacci numbers
C
      DO 510 I = 1,10
      CALL X25SDM (PORT, 3, IBUF, 25, 0, .TRUE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
510   CONTINUE

C Flush the buffered data to SVCRCV by calling X25SDM with 0 length
C
      CALL X25SDM (PORT, 3, IBUF, 0, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Calculate the first 128 square roots
C
      NUMBER = 1.0
      DO 600 I = 1,256,2
      XBUF(I) = NUMBER
      XBUF(I+1) = SQRT (NUMBER)
      NUMBER = NUMBER + 1.0
600   CONTINUE

C Send the identification string
C
      CALL X25SDM (PORT, 4, 9, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

      CALL X25SDM (PORT, 9, 30HNumbers and their Square Roots, 30,
     $             0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Send data array
C
      CALL X25SDM (PORT, 4, 0, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

      CALL X25SDM (PORT, 0, XBUF, 256, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Send control-Z to indicate end of data transmission
C
      CALL X25SDM (PORT, 4, "32, 1, 0, .FALSE., RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)

C Wait for confirmation of the interrupt we sent at the beginning
```

```
C
      CALL WTCFRM (PORT, 300)

C Clear virtual circuit and close port
C
      CALL X25CSC (PORT, "252, 0, 0, 0, 0, 0, RESULT)
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, CSTATE)
      CSTATE = 8

C Wait until clear request is confirmed
C
      CSTATE = ISTAT (PORT, CSTATE)

C Terminate port access
C
      CALL X25TPA (PORT, RESULT)

      STOP
      END

      SUBROUTINE ABORT (PORT, CODE, STATE)

C+
C DESCRIPTION        Terminate communication and abort the process.
C
C PARAMETERS PORT    Port number.
C                    CODE    Error code.
C                    STATE   Current port state.
C
C-

      EXTERNAL X25TPA
      INTEGER PORT, CODE, STATE

      WRITE (5,1000) CODE, STATE
1000  FORMAT (/,' * ERROR #',I2,', current port state ',I2,' *',/)

      CALL X25TPA (PORT, 0)

      STOP
      END

      SUBROUTINE WTCFRM (PORT, WAIT)

C+
C DESCRIPTION   Wait for interrupt confirmation within certain period.
C               After that, return, so we do not wait forever.
C
C PARAMETERS    PORT    Port number
C               WAIT    Amount of time in seconds to wait.
C-

      EXTERNAL X25RPS, IDLE

      INTEGER PORT, WAIT, RESULT
      LOGICAL PENDNG

C Read port status
C
      DO 100 I = 1,WAIT
      CALL X25RPS (PORT, 0, 0, 0, PENDNG, 0, 0, 0, RESULT)

C Check if the transmitted interrupt has been confirmed - if so, return
```

```
C
      IF (RESULT .NE. 0) CALL ABORT (PORT, RESULT, 5)
      IF (PENDNG .EQ. .FALSE.) RETURN

C Idle process for 1 second before checking the circuit status again
C
      CALL IDLE (1)
100   CONTINUE

      RETURN
      END

      INTEGER FUNCTION ISTAT (PORT, STATE)

C+
C DESCRIPTION   Read port status.
C
C PARAMETERS    PORT    Port number.
C               STATE   Current port state.
C
C RETURN        New port state when it is changes; or ERROR port state
C               if failed to read port status.
C-

      EXTERNAL X25RPS, IDLE
      INTEGER PORT, STATE, PSTATE, RESULT

C Check only port state, ignore other indicators
C
100   CALL X25RPS (PORT, 0, PSTATE, 0, 0, 0, 0, 0, RESULT)

C If failed to read port status, return port state ERROR
C
      IF (RESULT .EQ. 0) GOTO 200
      ISTAT = 10
RETURN

C If the port state has changes, return the new port state
C
200   IF (PSTATE .NE. STATE) GOTO 300

C Otherwise, idle the process for 5 seconds before checking
C the port state again
C
      CALL IDLE (5)
      GOTO 100

C Return new port state
C
300   ISTAT = PSTATE
      RETURN
      END
```

APPENDIX B

SAMPLE MACRO-10 PROGRAMS

This appendix contains sample MACRO-10 programs that communicate with each other through a PPSN. The programs are:

- SVCRCV, which accepts a Switched Virtual Circuit call and receives data sent through the circuit.
- SVCXMI, which initiates a Switched Virtual Circuit call and sends data through the circuit.

Section B.1 explains features of the two programs, while Section B.2 contains a listing of SVCRCV, and Section B.3 contains a listing of SVCXMI.

B.1 PROGRAM FEATURES

This section explains the set-up section that both programs use and gives hints related to individual features of each program.

B.1.1 Standard Set-up

Both programs include a standard set-up section that defines buffers and variables. The file UNV:X25SYM.UNV contains universal symbols such as virtual circuit port states (for example, XS%UND, XS%RUN), return codes (for example, XC%SUC, XC%PER) and other bit-mask symbols:

```
SEARCH  X25SYM
```

Because the TOPS-10 PSI Gateway Access routines use the push down stack for manipulation of data and other tasks, a MACRO-10 programmer should initialize the stack pointer to point to a push down list of at least 1000 (octal) words:

```
SIZE=1000
STACK:      BLOCK   SIZE
```

The chosen length of the data packets to be used in this example is 128 bytes. The length of the data block WSPACE, which is to be used by the X.25 Gateway Access routines as working space, is calculated as follows:

```
140 + (128/4) = 172
```

The size of the argument block ARGBLK is set at 10 so that it can be shared by all X.25 Gateway Access routines:

```
WSPSIZ=<128/4>+.PBSIZ
WSPACE:    BLOCK   WSPSIZ          ;Working area for the X.25 library
ARGBLK:    BLOCK   10.             ;Argument block
```

The set-up for the software interrupt system is standard (refer to the TOPS-10 Monitor Calls Manual for additional information). In this example, only interrupt channel zero is used. A priority level zero handler at SRVCKT is specified:

```
CH.X25=0                           ;Interrupt channel number
PSIARG:    EXP     .PCNSP          ;Condition, interrupt on NSP. events
           XWD     0,0             ;Offset into PSIVEC,,NOFLAGS
           XWD     0,0             ;Priority of 0
PSIVEC:    EXP     SRVCKT          ;New PC
           BLOCK   1               ;Old PC
           EXP     0               ;No Flags
           BLOCK   1               ;DECnet channel status word
```

B.1.2 Receiving Program

The program SVCRCV is written to act as the passive partner (slave) of the transmitting program, SVCXMI. SVCRCV starts by setting up the push down stack and the software interrupt system:

```
        SALL
        TITLE   SVCXMI  Communication Master Program
        SEARCH  UUOSYM,MACSYM,X25SYM

; Accumulators

S1=1                                    ;Scratch ACs
S2=2
S3=3
S4=4
T1=5                                    ;Temporary ACs
T2=6
SP=17

; Push Down Stack

SIZE=1000
STACK:  BLOCK   SIZE
        .
        .

; Interrupt System Storage Definitions

CH.X25=0                                ;Interrupt channel number
PSIARG: EXP     .PCNSP                  ;Condition, interrupt on NSP. events
        XWD     0,0                     ;Offset into PSIVEC,,NOFLAGS
        XWD     0,0                     ;Priority of 0
PSIVEC: EXP     SRVCKT                  ;New PC
        BLOCK   1                       ;Old PC
        EXP     0                       ;No Flags
        BLOCK   1                       ;DECnet channel status word

START:  RESET                           ;Reset the world
        MOVE    SP,[IOWD SIZE,STACK]    ;Set up push down stack

PSIINI: MOVEI   S1,PSIVEC
        PIINI.  S1,                     ;Initialize software interrupt system
         HALT                           ; Failed
        MOVE    T1,[PS.FOF!PS.FAC+PSIARG]
        PISYS.  T1,                     ;Keep interrupt system off temporarily
         HALT                           ; Failed
        .
        .
```

When you specify a positive number as an interrupt channel in the argument block of the subroutines X%ISC, X%OPC, or X%WIC, the number does not have any particular significance in itself. It is simply a flag that tells the Gateway Access Library to enable software interrupts for that X.25 circuit. All DECnet and X.25 programmed interrupts come through a single PSI vector. You cannot assign a different vector to each X.25 circuit as you can for normal TOPS-10 I/O.

When you enable the software interrupts for an X.25 circuit, the port number, which is returned by the subroutines X%ISC, X%OPC, or X%WIC, is also the software interrupt channel number. The port number can be used to determine which circuit the interrupt is for when it occurs.

When the system grants an interrupt, you can determine which circuit the interrupt is for by looking at the status word of the interrupt control block (fourth word). It will contain the interrupt channel number (port number) in the right half and the link status (DECnet format) in the left half. If the status word is zero, you should ignore the interrupt.

The Gateway Access subroutines are linked with your program image, therefore, it is possible for an interrupt to occur while your program executes a subroutine call. Use the Software Interrupt System carefully. You must insure that the system will grant no interrupts until all the necessary data structures have been initialized, control has been returned to the user code, and no interrupts have been lost.

The Gateway Access subroutines save the context of all sixteen registers before using them. When an interrupt occurs, you must verify your register's contents to make sure they are not interrupted.

An X.25 circuit must be established before interrupts can be taken on that link. However, if the Software Interrupt System is not enabled before the program calls a Gateway Access subroutine to establish the circuit, interrupts can be lost as they are granted at the first possible instance. To avoid this problem, you should initialize the Software Interrupt System first, before initiating any X.25 circuits to prevent interrupts being lost. Turn the Software Interrupt System off just before calling the subroutines X%ISC, X%OPC, or X%WIC and turn it back on after program control is returned to the user code.

The Gateway Access subroutines X%ISC, X%OPC, and X%WIC set up the conditions that would trigger software interrupts on an X.25 circuit. You should not change that setting during the "life" of that circuit. Use the routine X%RPS (Read Port Status) to find out about events on the virtual circuit. Use the software interrupt system whenever possible. Normally, the Software Interrupt System enables a program to run more efficiently.

SVCRCV declares itself available to receive an incoming call from the X.25 gateway node by calling the routine X%WIC (Wait Incoming Call):

```
        .
        .
ARGINI: SETZM   ARGBLK                  ;Initialize entire argument block area
        MOVE    T1,[ARGBLK,,ARGBLK+1]
        BLT     T1,ARGBLK+9.

X25ISC: MOVEI   T1,XS%UND
        MOVEM   T1,STATE                ;Set port state to UNDEFINED
        MOVE    T1,[CH.X25,,WSPACE]
        MOVEM   T1,ARGBLK               ;Interrupt channel and work area
        MOVE    T1,[POINT 7,[ASCIZ/TELENET/]]
        MOVEM   T1,ARGBLK+2             ;Network name
        MOVE    T1,[POINT 7,[ASCIZ/X25-GATE/]]
        MOVEM   T1,ARGBLK+3             ;Access password
        MOVE    T1,[POINT 7,[ASCIZ/311060700153/]]
        MOVEM   T1,ARGBLK+4             ;Destination DTE address
        MOVEI   S1,ARGBLK
        CALL    X%ISC##
        MOVE    T1,ARGBLK+1             ;Get return code
        CAIE    T1,XC%SUC               ;Is it successful ?
         EXIT   1,                      ; No, terminate
        MOVEI   T1,XS%CAG               ;Yes,
        MOVEM   T1,STATE                ;Change port state to CALLING

DISMIS: MOVE    T1,[PS.FON]
        PISYS.  T1,                     ;Turn interrupt system on
         EXIT   1,                      ; Failed
        MOVEI   T1,0
DSM.1:  HIBER   T1,                     ;Dismiss the process
         EXIT   1,                      ; Failed
        JRST    DSM.1
        .
        .
```

After establishing itself as a server, the process dismisses itself and waits for an incoming call from the network. While your program is waiting for an X.25 incoming call, you should use the HIBER UUO with a timer of 0 seconds (hibernate indefinitely) instead of the SLEEP UUO. The HIBER UUO requires less overhead in a program that may wait for a long period of time.

When an event occurs on the virtual circuit, the monitor transfers control to the interrupt routine SRVCKT, which is specified in the interrupt control block PSIVEC. The first thing the interrupt handler routine does is check the current status of the virtual circuit by calling the routine X%RPS. The routine then transfers control to the appropriate routine, which processes the port state transitions or events (for example, incoming data):

```
        .
        .
SRVCKT: PUSH    SP,S1                   ;Save registers
        PUSH    SP,S2
        PUSH    SP,T1
        PUSH    SP,T2

        SETZM   ARGBLK+1                ;Initialize argument block entries
        MOVE    T1,[ARGBLK+1,,ARGBLK+2]
        BLT     T1,ARGBLK+9.
        MOVEI   S1,ARGBLK               ;Pass access argument block
        CALL    X%RPS##                 ;Check port state
        MOVE    T1,ARGBLK+1             ;Get return code
        CAIE    T1,XC%SUC               ;Is it successful ?
         JRST   X25TPA                  ;No, terminate
        MOVE    T1,STATE                ;Get old port state
        HRRZ    T2,ARGBLK+2             ;Get new port state
        CAIN    T1,XS%CAG               ;From CALLING state
         JRST   [CAIN   T2,XS%CAG       ;Remain in CALLING state
                  JRST  NOTHNG          ;Do nothing
                 CAIN   T2,XS%RUN       ;Changes to RUNNING state
                  JRST  X25SDM          ;Transmit data to the network
                 JRST   X25TPA]         ;Otherwise, terminate
        CAIN    T1,XS%RUN               ;From RUNNING state
         JRST   [CAIN   T2,XS%RUN       ;Remain in RUNNING state
                  JRST  NOTHNG          ;Do nothing
                 JRST   X25TPA]         ;Otherwise, terminate
        JRST    X25TPA                  ;Don't want to handle other states

NOTHNG: POP     SP,T2                   ;Restore registers
        POP     SP,T1
        POP     SP,S2
        POP     SP,S1
        DEBRK.
        .
        .
```

The current port state (which has been maintained by the program) and the new one (newly obtained from the routine X%RPS) are examined to determine the port state transition. In the above example, the port state transition from XS%LSN (LISTENING) to XS%CAD (CALLED), indicates that the program has received an incoming call from the network. The program proceeds to accept the incoming call:

```
        .
        .
X25AIC: SETZM   ARGBLK+1                ;Initialize argument block entries
        MOVE    T1,[ARGBLK+1,,ARGBLK+2]
        BLT     T1,ARGBLK+9.
        MOVEI   S1,ARGBLK               ;Accept incoming call unconditionally
        CALL    X%AIC##
        MOVE    T1,ARGBLK+1             ;Get return code
        CAIE    T1,XC%SUC               ;Is it successful ?
         JRST   X25TPA                  ; No, terminate
        MOVEI   T1,XS%RUN               ;Yes,
        MOVEM   T1,STATE                ;Change port state to RUNNING
        JRST    NOTHNG                  ;Done servicing interrupt
        .
        .
```

After accepting the incoming call, the program has updated its copy of the current port state to XS%RUN (RUNNING). Note that there are other subroutines that change the port state upon successful execution. When one of these subroutines has been successfully executed, you do not need to execute X%RPS to determine the port state. For example:

X%RVC (RESET VIRTUAL CIRCUIT), which changes the port state from UNSYNC to RUNNING.

You can detect events on the virtual circuit by examining word ARGBLK+3 of the argument block that is returned by X%RPS. The above example tests the bit XM%DAT to determine if there is a data packet to be read from the virtual circuit.

The following section of the program receives data packets and displays them on the control terminal:

```
        .
        .
X25RDM: SETZM   ARGBLK+1                ;Initialize argument block entries
        MOVE    T1,[ARGBLK+1,,ARGBLK+2]
        BLT     T1,ARGBLK+9.

RDM.1:  MOVEI   T1,128.
        MOVEM   T1,ARGBLK+2             ;Maximum packet size
        MOVE    T1,[POINT 7,BUFFER]
        MOVEM   T1,ARGBLK+3             ;Destination buffer
        MOVEI   S1,ARGBLK
        CALL    X%RDM##                 ;Read one packet
        MOVE    T1,ARGBLK+1             ;Get return code
        CAIE    T1,XC%SUC               ;Is it successful ?
         JRST   [TXNE   T1,XC%NDA       ; No, is it No Data Available error ?
                  JRST  NOTHNG          ;  Yes, dismiss interrupt
                 JRST   X25TPA]         ;No, it is fatal error
        MOVE    T1,ARGBLK+3             ;Get pointer to last received byte
        MOVEI   T2,0
        IDPB    T2,T1                   ;Make ASCIZ string
        OUTSTR  BUFFER                  ;Display text at terminal
        JRST    RDM.1                   ;Read the next packet
        .
        .
```

Note that while reading data from the network, the program ignores the value of bit XM%MOR, which represents the more bit of the received data packet. (XM%MOR is returned in word ARGBLK+2 of the argument block by routine X%RDM.) This is because the more bit of a data packet should only be used in the context of logical streams of received data. The more bit is not a reliable indicator of additional data packets being available on the virtual circuit at that moment. The content of word ARGBLK+1 (return code) of the argument block will indicate when there is no more data to be read (bit XC%NDA is set).

When your program is notified by the interrupt system and the routine X%RPS to handle the incoming data, do not call X%RPS to obtain the value of bit XM%DAT before reading each data packet, unless you are processing other events simultaneously. Use the routine X%RDM to read data continuously until X%RDM returns the return code which indicates that there is no more data to be read (bit XC%NDA is set) or there are other error conditions.

Finally, when error conditions are encountered, the program terminates by clearing the virtual circuit and halting itself:

```
                .
                .
X25TPA:         MOVEI   S1,ARGBLK
                CALL    X%TPA##                 ;Terminate port access
                EXIT    1,                      ;Stop program
                END     START                   ;End Of Program
```

B.1.3 Transmitting Program

The program SVCXMI is the active partner (master) of the receiving program, SVCRCV. This program initiates all of the operations; that is, it initiates the Switched Virtual Circuit call and the data transfer. The program interacts with the user to acquire data to be transferred to the receiving program.

The program first initializes the process environment and the software interrupt system:

```
        SALL
        TITLE   SVCRCV  Communication Slave Program
        SEARCH  UUOSYM,MACSYM,X25SYM

; Accumulators

S1=1                                    ;Scratch ACs
S2=2
S3=3
S4=4
T1=5                                    ;Temporary ACs
T2=6
SP=17

; Push Down Stack

SIZE=1000
STACK:  BLOCK   SIZE
        .
        .

; Interrupt System Storage Definitions

CH.X25=0                                ;Interrupt channel number
PSIARG: EXP     .PCNSP                  ;Condition, interrupt on NSP. events
        XWD     0,0                     ;Offset into PSIVEC,,NOFLAGS
        XWD     0,0                     ;Priority of 0
PSIVEC: EXP     SRVCKT                  ;New PC
        BLOCK   1                       ;Old PC
        EXP     0                       ;No Flags
        BLOCK   1                       ;DECnet channel status word

START:  RESET                           ;Reset the world
        MOVE    SP,[IOWD SIZE,STACK]    ;Set up push down stack

PSIINI: MOVEI   S1,PSIVEC
        PIINI.  S1,                     ;Initialize software interrupt system
         HALT                           ; Failed
        MOVE    T1,[PS.FOF!PS.FAC+PSIARG]
        PISYS.  T1,                     ;Keep interrupt system off temporarily
         HALT                           ; Failed
        .
        .
```

The program proceeds to initiate the Switched Virtual Circuit:

```
        .
        .
ARGINI: SETZM   ARGBLK                  ;Initialize entire argument block area
        MOVE    T1,[ARGBLK,,ARGBLK+1]
        BLT     T1,ARGBLK+9.

X25ISC: MOVEI   T1,XS%UND
        MOVEM   T1,STATE                ;Set port state to UNDEFINED
        MOVE    T1,[CH.X25,,WSPACE]
        MOVEM   T1,ARGBLK               ;Interrupt channel and work area
        MOVE    T1,[POINT 7,[ASCIZ/TELENET/]]
        MOVEM   T1,ARGBLK+2             ;Network name
        MOVE    T1,[POINT 7,[ASCIZ/X25-GATE/]]
        MOVEM   T1,ARGBLK+3             ;Access password
        MOVE    T1,[POINT 7,[ASCIZ/311060700153/]]
        MOVEM   T1,ARGBLK+4             ;Destination DTE address
        MOVEI   S1,ARGBLK
        CALL    X%ISC##
        MOVE    T1,ARGBLK+1             ;Get return code
        CAIE    T1,XC%SUC               ;Is it successful ?
         EXIT   1,                      ; No, terminate
        MOVEI   T1,XS%CAG               ;Yes,
        MOVEM   T1,STATE                ;Change port state to CALLING

DISMIS: MOVE    T1,[PS.FON]
        PISYS.  T1,                     ;Turn interrupt system on
         EXIT   1,                      ; Failed
        MOVEI   T1,0
DSM.1:  HIBER   T1,                     ;Dismiss the process
         EXIT   1,                      ; Failed
        JRST    DSM.1
        .
        .
```

When an event occurs on the virtual circuit, the monitor transfers control to the interrupt routine SRVCKT, which is specified in the interrupt control block PSIVEC. The routine SRVCKT serves as a port state driven machine. Depending on the port state of the virtual circuit at the time of the interrupt, the program transfers control to an associated routine.

```
        .
        .
SRVCKT: PUSH    SP,S1                   ;Save registers
        PUSH    SP,S2
        PUSH    SP,T1
        PUSH    SP,T2

        SETZM   ARGBLK+1                ;Initialize argument block entries
        MOVE    T1,[ARGBLK+1,,ARGBLK+2]
        BLT     T1,ARGBLK+9.
        MOVEI   S1,ARGBLK               ;Pass access argument block
        CALL    X%RPS##                 ;Check port state
        MOVE    T1,ARGBLK+1             ;Get return code
        CAIE    T1,XC%SUC               ;Is it successful ?
         JRST   X25TPA                  ;No, terminate
        MOVE    T1,STATE                ;Get old port state
        HRRZ    T2,ARGBLK+2             ;Get new port state
        CAIN    T1,XS%CAG               ;From CALLING state
         JRST   [CAIN   T2,XS%CAG       ;Remain in CALLING state
                  JRST  NOTHNG          ;Do nothing
                 CAIN   T2,XS%RUN       ;Changes to RUNNING state
                  JRST  X25SDM          ;Transmit data to the network
                 JRST   X25TPA]         ;Otherwise, terminate
        CAIN    T1,XS%RUN               ;From RUNNING state
         JRST   [CAIN   T2,XS%RUN       ;Remain in RUNNING state
                  JRST  NOTHNG          ;Do nothing
                 JRST   X25TPA]         ;Otherwise, terminate
        JRST    X25TPA                  ;Don't want to handle other states

NOTHNG: POP     SP,T2                   ;Restore registers
        POP     SP,T1
        POP     SP,S2
        POP     SP,S1
        DEBRK.
        .
        .
```

Upon an interrupt, the program checks the current port state by calling routine X%RPS (Read Port Status) and transfers control to the appropriate routine. The current port state and the new one are examined to determine the port state transition. In the above example, the port state transition from XS%CAG (CALLING) to XS%RUN (RUNNING), indicates that the switched virtual circuit has been established successfully. The program now can proceed to transmit data to the network:

```
          .
          .
X25SDM:   MOVEI   T1,XS%RUN
          MOVEM   T1,STATE                ;Update port state to RUNNING
          OUTSTR  HELP

SDM.1:    OUTSTR  PROMPT
          MOVE    T1,[POINT 7,BUFFER]     ;Receiving buffer
          SETZ    T2,                     ;Reset counter

SDM.2:    INCHWL  S1                      ;Get one character from terminal
          CAIN    S1,32                   ;End of text ?
           JRST   X25TPA                  ; Yes, terminate circuit
          IDPB    S1,T1                   ;Stash character away
          AOS     T2                      ;Update counter
          CAIE    S1,12                   ;End of text line ?
           JRST   SDM.2                   ; No, continue

SDM.3:    SETZM   ARGBLK+1                ;Initialize argument block entries
          MOVE    T1,[ARGBLK+1,,ARGBLK+2]
          BLT     T1,ARGBLK+9.
          CAILE   T2,128.                 ;Set the maximum length of the string
           MOVEI  T2,128.                 ; Truncate the text to 128 characters
          MOVEM   T2,ARGBLK+2
          MOVE    T1,[POINT 7,BUFFER]
          MOVEM   T1,ARGBLK+3             ;Get pointer to the text string
          MOVEI   S1,ARGBLK
          CALL    X%SDM##                 ;Send the string to the network
          MOVE    T1,ARGBLK+1             ;Get the return code
          CAIE    T1,XC%SUC               ;Is it successful ?
           JRST   [TXNN   T1,XC%PER       ;No, then is it procedure error ?
                     JRST  SDM.1          ;No, it is non-fatal error
                    JRST   X25TPA]        ;Yes, terminate
          JRST    SDM.1                   ;Get next string from terminal
          .
          .
```

The program prompts for user data from the controlling terminal and sends the text to the receiving program. The program terminates when you enter a CTRL/Z:

```
          .
          .
X25TPA:   MOVEI   S1,ARGBLK
          CALL    X%TPA##                 ;Terminate port access
          EXIT    1,                      ;Stop program
          END     START                   ;End Of Program
```

B.2 LISTING OF SAMPLE PROGRAM SVCRCV.MAC

```
        SALL
        TITLE       SVCRCV  Communication Slave Program
        SEARCH      UUOSYM,MACSYM,X25SYM

; Accumulators

S1=1                                            ;Scratch ACs
S2=2
S3=3
S4=4
T1=5                                            ;Temporary ACs
T2=6
SP=17

; Push Down Stack

SIZE=1000
STACK:      BLOCK   SIZE

; X.25 Access Argument Blocks

WSPSIZ=<128./4>+.PBSIZ
WSPACE:     BLOCK   WSPSIZ                      ;Working area for the X.25 library
ARGBLK:     BLOCK   10.                         ;Argument block
BUFFER:     BLOCK   32.                         ;Incoming data buffer
STATE:      BLOCK   1                           ;Port state

; Interrupt System Storage Definitions

CH.X25=0                                        ;Interrupt channel number
PSIARG:     EXP     .PCNSP                      ;Condition, interrupt on NSP. events
            XWD     0,0                         ;Offset into PSIVEC,,NOFLAGS
            XWD     0,0                         ;Priority of 0
PSIVEC:     EXP     SRVCKT                      ;New PC
            BLOCK   1                           ;Old PC
            EXP     0                           ;No Flags
            BLOCK   1                           ;DECnet channel status word

START:      RESET                               ;Reset the world
            MOVE    SP,[IOWD SIZE,STACK]        ;Set up push down stack

PSIINI:     MOVEI   S1,PSIVEC
            PIINI.  S1,                         ;Initialize software interrupt system
             HALT                               ; Failed
            MOVE    T1,[PS.FOF!PS.FAC+PSIARG]
            PISYS.  T1,                         ;Keep interrupt system off temporarily
             HALT                               ; Failed

ARGINI:     SETZM   ARGBLK                      ;Initialize the argument block
            MOVE    T1,[ARGBLK,,ARGBLK+1]
            BLT     T1,ARGBLK+9.

X25WIC:     MOVEI   T1,XS%UND
            MOVEM   T1,STATE                    ;Set port state to UNDEFINED
            MOVE    T1,[CH.X25,,WSPACE]
            MOVEM   T1,ARGBLK                   ;Interrupt channel and work area
            MOVE    T1,[POINT 7,[ASCIZ/EXAMPLE/]]
            MOVEM   T1,ARGBLK+2                 ;DECnet object name
            MOVEI   S1,ARGBLK                   ;Pass access argument block
            CALL    X%WIC##                     ;Wait Incoming Call
            MOVE    T1,ARGBLK+1                 ;Get return code
            CAIE    T1,XC%SUC                   ;Is it successful ?
             EXIT   1,                          ; No, terminate
            MOVEI   T1,XS%LSN                   ;Yes,
            MOVEM   T1,STATE                    ;Change port state to LISTENING

DISMIS:     MOVE    T1,[PS.FON]
            PISYS.  T1,                         ;Turn interrupt system on
             EXIT   1,                          ; Failed
            MOVEI   T1,0
```

```
DSM.1:      HIBER   T1,                     ;Dismiss the process
             EXIT   1,                      ; Failed
            JRST    DSM.1

SRVCKT:     PUSH    SP,S1                   ;Save registers
            PUSH    SP,S2
            PUSH    SP,T1
            PUSH    SP,T2

            SETZM   ARGBLK+1                ;Initialize argument block entries
            MOVE    T1,[ARGBLK+1,,ARGBLK+2]
            BLT     T1,ARGBLK+9.
            MOVEI   S1,ARGBLK               ;Pass access argument block
            CALL    X%RPS##                 ;Check current port status
            MOVE    T1,ARGBLK+1             ;Get return code
            CAIE    T1,XC%SUC               ;Is it successful ?
             JRST   X25TPA                  ; No, terminate
            MOVE    T1,STATE                ;Get old port state
            HRRZ    T2,ARGBLK+2             ;Get new port state
            CAIN    T1,XS%LSN               ;From LISTENING state
             JRST   [CAIN   T2,XS%LSN       ; Remain in LISTENING state
                      JRST  NOTHNG          ;  Check the port state again
                     CAIN   T2,XS%CAD       ; Changes to CALLED state
                      JRST  X25AIC          ;  Accept incoming call
                     JRST   X25TPA]         ; Otherwise, terminate
            CAIN    T1,XS%RUN               ;From RUNNING state
             JRST   [CAIE   T2,XS%RUN       ; Changes to any other states
                      JRST  X25TPA          ;  Terminate
                     MOVE   T1,ARGBLK+3     ; Get the port status word
                     TXNE   T1,XM%DAT       ; Check incoming data indicator
                      JRST  X25RDM          ;  Yes, there is a packet to be read
                     JRST   NOTHNG]         ; No, done servicing interrupt
            JRST    X25TPA                  ;Don't want to handle other states

X25AIC:     SETZM   ARGBLK+1                ;Initialize argument block entries
            MOVE    T1,[ARGBLK+1,,ARGBLK+2]
            BLT     T1,ARGBLK+9.
            MOVEI   S1,ARGBLK               ;Accept incoming call unconditionally
            CALL    X%AIC##
            MOVE    T1,ARGBLK+1             ;Get return code
            CAIE    T1,XC%SUC               ;Is it successful ?
             JRST   X25TPA                  ; No, terminate
            MOVEI   T1,XS%RUN               ;Yes,
            MOVEM   T1,STATE                ;Change port state to RUNNING
            JRST    NOTHNG                  ;Done servicing interrupt

X25RDM:     SETZM   ARGBLK+1                ;Initialize argument block entries
            MOVE    T1,[ARGBLK+1,,ARGBLK+2]
            BLT     T1,ARGBLK+9.

RDM.1:      MOVEI   T1,128.
            MOVEM   T1,ARGBLK+2             ;Maximum packet size
            MOVE    T1,[POINT 7,BUFFER]
            MOVEM   T1,ARGBLK+3             ;Destination buffer
            MOVEI   S1,ARGBLK
            CALL    X%RDM##                 ;Read one packet
            MOVE    T1,ARGBLK+1             ;Get return code
            CAIE    T1,XC%SUC               ;Is it successful ?
             JRST   [TXNE   T1,XC%NDA       ; No, is it No Data Available error ?
                      JRST  NOTHNG          ;  Yes, dismiss interrupt
                     JRST   X25TPA]         ;No, it is fatal error
            MOVE    T1,ARGBLK+3             ;Get pointer to last received byte
            MOVEI   T2,0
            IDPB    T2,T1                   ;Make ASCIZ string
            OUTSTR  BUFFER                  ;Display text at terminal
            JRST    RDM.1                   ;Read the next packet

NOTHNG:     POP     SP,T2                   ;Restore registers
            POP     SP,T1
            POP     SP,S2
            POP     SP,S1
            DEBRK.

X25TPA:     MOVEI   S1,ARGBLK
            CALL    X%TPA##                 ;Terminate port access
            EXIT    1,                      ;Stop program
            END     START                   ;End Of Program
```

B.3 LISTING OF SAMPLE PROGRAM SVCXMI.MAC

```
        SALL
        TITLE   SVCXMI  Communication Master Program
        SEARCH  UUOSYM,MACSYM,X25SYM

; Accumulators

S1=1                                    ;Scratch ACs
S2=2
S3=3
S4=4
T1=5                                    ;Temporary ACs
T2=6
SP=17

; Push Down Stack

SIZE=1000
STACK:  BLOCK   SIZE

; X.25 Access Argument Blocks

WSPSIZ=<128./4>+.PBSIZ
WSPACE: BLOCK   WSPSIZ                  ;Working area for the X.25 library
ARGBLK: BLOCK   ^D10                    ;Argument block
BUFFER: BLOCK   ^D32                    ;Transmitted data buffer
STATE:  BLOCK   1                       ;Port state
PROMPT: ASCIZ   /> /
HELP:   ASCIZ   /Enter text:
/

; Interrupt System Storage Definitions

CH.X25=0                                        ;Interrupt channel number
PSIARG: EXP     .PCNSP                  ;Condition, interrupt on NSP. events
        XWD     0,0                     ;Offset into PSIVEC,,NOFLAGS
        XWD     0,0                     ;Priority of 0
PSIVEC: EXP     SRVCKT                  ;New PC
        BLOCK   1                       ;Old PC
        EXP     0                       ;No Flags
        BLOCK   1                       ;DECnet channel status word

START:  RESET                           ;Reset the world
        MOVE    SP,[IOWD SIZE,STACK]    ;Set up push down stack

PSIINI: MOVEI   S1,PSIVEC
        PIINI.  S1,                     ;Initialize software interrupt system
         HALT                           ; Failed
        MOVE    T1,[PS.FOF!PS.FAC+PSIARG]
        PISYS.  T1,                     ;Keep interrupt system off temporarily
         HALT                           ; Failed

ARGINI: SETZM   ARGBLK                  ;Initialize entire argument block area
        MOVE    T1,[ARGBLK,,ARGBLK+1]
        BLT     T1,ARGBLK+9.

X25ISC: MOVEI   T1,XS%UND
        MOVEM   T1,STATE                ;Set port state to UNDEFINED
        MOVE    T1,[CH.X25,,WSPACE]
        MOVEM   T1,ARGBLK               ;Interrupt channel and work area
        MOVE    T1,[POINT 7,[ASCIZ/TELENET/]]
        MOVEM   T1,ARGBLK+2             ;Network name
        MOVE    T1,[POINT 7,[ASCIZ/X25-GATE/]]
        MOVEM   T1,ARGBLK+3             ;Access password
        MOVE    T1,[POINT 7,[ASCIZ/311060700153/]]
        MOVEM   T1,ARGBLK+4             ;Destination DTE address
        MOVEI   S1,ARGBLK
        CALL    X%ISC##
        MOVE    T1,ARGBLK+1             ;Get return code
        CAIE    T1,XC%SUC               ;Is it successful ?
         EXIT   1,                      ; No, terminate
        MOVEI   T1,XS%CAG               ;Yes,
        MOVEM   T1,STATE                ;Change port state to CALLING
```

```
DISMIS: MOVE    T1,[PS.FON]
        PISYS.  T1,                     ;Turn interrupt system on
         EXIT   1,                      ; Failed
        MOVEI   T1,0
DSM.1:  HIBER   T1,                     ;Dismiss the process
         EXIT   1,                      ; Failed
        JRST    DSM.1

SRVCKT: PUSH    SP,S1                   ;Save registers
        PUSH    SP,S2
        PUSH    SP,T1
        PUSH    SP,T2

        SETZM   ARGBLK+1                ;Initialize argument block entries
        MOVE    T1,[ARGBLK+1,,ARGBLK+2]
        BLT     T1,ARGBLK+9.
        MOVEI   S1,ARGBLK               ;Pass access argument block
        CALL    X%RPS##                 ;Check port state
        MOVE    T1,ARGBLK+1             ;Get return code
        CAIE    T1,XC%SUC               ;Is it successful ?
         JRST   X25TPA                  ;No, terminate
        MOVE    T1,STATE                ;Get old port state
        HRRZ    T2,ARGBLK+2             ;Get new port state
        CAIN    T1,XS%CAG               ;From CALLING state
         JRST   [CAIN   T2,XS%CAG       ;Remain in CALLING state
                  JRST  NOTHNG          ;Do nothing
                 CAIN   T2,XS%RUN       ;Changes to RUNNING state
                  JRST  X25SDM          ;Transmit data to the network
                 JRST   X25TPA]         ;Otherwise, terminate
        CAIN    T1,XS%RUN               ;From RUNNING state
         JRST   [CAIN   T2,XS%RUN       ;Remain in RUNNING state
                  JRST  NOTHNG          ;Do nothing
                 JRST   X25TPA]         ;Otherwise, terminate
        JRST    X25TPA                  ;Don't want to handle other states

NOTHNG: POP     SP,T2                   ;Restore registers
        POP     SP,T1
        POP     SP,S2
        POP     SP,S1
        DEBRK.

X25SDM: MOVEI   T1,XS%RUN
        MOVEM   T1,STATE                ;Update port state to RUNNING
        OUTSTR  HELP

SDM.1:  OUTSTR  PROMPT
        MOVE    T1,[POINT 7,BUFFER]     ;Receiving buffer
        SETZ    T2,                     ;Reset counter

SDM.2:  INCHWL  S1                      ;Get one character from terminal
        CAIN    S1,32                   ;End of text ?
         JRST   X25TPA                  ; Yes, terminate circuit
        IDPB    S1,T1                   ;Stash character away
        AOS     T2                      ;Update counter
        CAIE    S1,12                   ;End of text line ?
         JRST   SDM.2                   ; No, continue

SDM.3:  SETZM   ARGBLK+1                ;Initialize argument block entries
        MOVE    T1,[ARGBLK+1,,ARGBLK+2]
        BLT     T1,ARGBLK+9.
        CAILE   T2,128.                 ;Set the maximum length of the string
         MOVEI  T2,128.                 ; Truncate the text to 128 characters
        MOVEM   T2,ARGBLK+2
        MOVE    T1,[POINT 7,BUFFER]
        MOVEM   T1,ARGBLK+3             ;Get pointer to the text string
        MOVEI   S1,ARGBLK
        CALL    X%SDM##                 ;Send the string to the network
        MOVE    T1,ARGBLK+1             ;Get the return code
        CAIE    T1,XC%SUC               ;Is it successful ?
         JRST   [TXNN   T1,XC%PER       ;No, then is it procedure error ?
                  JRST  SDM.1           ;No, it is non-fatal error
                 JRST   X25TPA]         ;Yes, terminate
        JRST    SDM.1                   ;Get next string from terminal

X25TPA: MOVEI   S1,ARGBLK
        CALL    X%TPA##                 ;Terminate port access
        EXIT    1,                      ;Stop program
        END     START                   ;End Of Program
```

APPENDIX C

BIBLIOGRAPHY

The following lists CCITT recommendations relevant to users of the TOPS-10 PSI software and the CCITT publications in which those recommendations are discussed. The information in this appendix is subject to change. For more information, contact the CCITT.

Recommendation	Document
X.3	Document AP VII-No. 6-E
X.25	Document AP VII-No. 7
X.28, X.29	Document AP VII-No. 8

The following document contains a complete and unabridged form of the key CCITT recommendations as they appear in Fascile VIII.2 of the Yellow Book of the Consultative Committee for International Telegraph and Telephone, 1981 edition.

The X.25 Protocol and Seven Other Key CCITT Recommendations: X.1, X.2, X.3, X.21 bis, X.28, and X.29. Lifetime Learning Publications. Belmont, California. 1981

APPENDIX D

ERROR MESSAGES DISPLAYED BY X29SRV

Error Message	Section(s) in Which Message Is Discussed
BREAK	5.6
CONTINUE	5.6
CONNECTION IS NOT SUPPORTED	5.5.3
DISCONNECTING	5.5.2, 5.5.3
FAILED TO CONNECT	5.5.3
HOST SYSTEM DISCONNECTED	5.5.5
TIMED OUT	5.5.3
UNDEFINED X.3 PARAMETER SET	5.5.3, 5.5.4

APPENDIX E

GLOSSARY

Bilateral Closed User Group (BCUG)

An optional PPSN facility that restricts a pair of DTEs to communicating with each other. The basic BCUG also prevents this pair from accessing or being accessed by other DTEs. Additions to the BCUG facility allow one or both of the DTEs to access or be accessed by DTEs outside the group. These additions are known as BCUG with Outgoing Access and BCUG with Incoming Access respectively.

CCITT

Comite Consultatif International Telegraphique et Telephonique. An international advisory committee that sets international communications usage standards.

Channel

A logical path between a DTE and a DCE over which data is transmitted. A unique reference number called a Logical Channel Number (LCN) identifies each channel.

Character Mode DTE

A DTE that is unable to handle data in packet form. This DTE must interface through a Packet Assembly/Disassembly (PAD) facility to connect to a PPSN. Also known as a Remote X.29 Terminal.

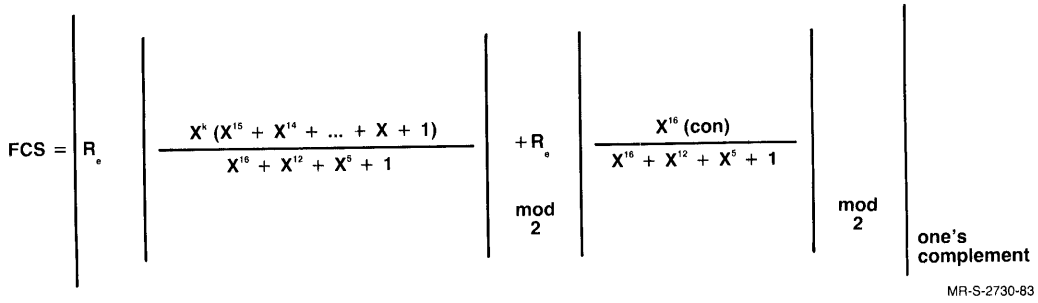
Closed User Group (CUG)

An optional PPSN facility that restricts two or more DTEs in the same group to communicating with each other. The basic CUG also prevents these DTEs from accessing or being accessed by other DTEs outside the group. Additions to the basic CUG facility allow one or more DTEs to access or be accessed by DTEs outside the group. These additions are known as CUG with Outgoing Access and CUG with Incoming Access respectively.

Data Circuit-terminating Equipment (DCE)	A CCITT X.25 term referring to the network equipment that provides functions to establish, maintain and terminate a connection. A DCE also handles the signal conversion and coding between the data terminal equipment and the network. The switching exchange of the network to which DTEs are connected. (In non-X.25 usage, the term is synonymous with 'modem'.)
Data Terminal Equipment (DTE)	A CCITT term referring to the user's equipment (computer or terminal) connected to a DCE on a packet switching network for the purpose of sending and/or receiving data.
DCE	See Data Circuit-terminating Equipment.
DECnet	The collective name for the software and hardware products that allow various DIGITAL operating systems to be interconnected to form computer networks. A network is a configuration of two or more independent computer systems linked together to share resources and/or exchange information.
DTE	See Data Terminal Equipment.
Duplex	Simultaneous, independent transmission in both directions. Also referred to as full-duplex.
Facility	A service or mode of operating that a PPSN is able to provide for a user upon subscription and/or request, for example, fast select or reverse charging.
Fast Select	An optional PPSN facility that allows a DTE to include a user data field of up to 128 bytes when setting up a virtual circuit.
Flag Sequence	A series of ones and zeros that indicates the start and end of a frame.
Flow Control	The mechanism which ensures that the sending station does not overrun the receiving station with more packets that it can accept.
Flow Control Parameter Negotiation	A process that allows selection of packet sizes and window sizes in each direction of a particular virtual circuit.
Frame	A unit delimited by flags that includes a header, used by the link level to exchange packets as well as control and error information between the DTE and the DCE.

Frame Checking Sequence (FCS)

16-bit, error check polynomial which verifies that the bit content of a frame is the same before and after transmission.



where k = number of bits in the frame not including the flags and FCS itself and con = the contents of the frame not including the flags and the FCS itself.

Full-Duplex

See Duplex.

Gateway

The connection between two individual packet switching networks. The connection provides a link through which a DTE can communicate with a DTE on a different network. A gateway is covered in CCITT Recommendation X.75.

Half-Duplex

A circuit designed for transmission in either direction but not both directions simultaneously.

Header

The control information before a message text; for example, source or destination code, priority, or packet or frame identification.

Incoming Calls Barred

An optional PPSN facility that prevents a DTE from accepting any calls.

Local DTE

A frame of reference; the DTE at which the user is located.

Logical Channel

A logical link between a DTE and its DCE. The physical communications line between a DTE and DCE is divided into a set of logical channels.

Logical Channel Number (LCN)

A unique reference number that identifies a logical channel. A DTE recognizes a virtual circuit by its associated LCN.

Message

A communication, prepared for information interchange in a form suitable for passage through the interchange medium. It includes:

- All portions of the communications such as machine sensible controls
- An indication of the start of the message and the end of the message
- A header containing routing and other information, one or more texts containing the originator-to-addressee communication(s), and the end of text indicator

In packet switching, a message can be segmented into several packets to traverse the network, or in some circumstances several messages can be carried in one packet.

Modem (Modulator-Demodulator)

A device that translates digital signals (electrical impulses) generated by a computer into analogue signals (tones) that can be transmitted over telephone lines, and vice versa.

Non-packet-mode DTE

See Character Mode DTE.

Non-standard Default Packet Size

An optional PPSN facility that permits a DTE to specify a default packetsize that is different from the PPSN's default.

Non-standard Default Window Size

An optional PPSN facility that permits a DTE to specify a default window size that is different from the PPSN's default.

Octet

A group of eight bits; a byte.

One-way Logical Channel Incoming

An optional PPSN facility that prevents a particular logical channel from handling outgoing calls.

One-way Logical Channel Outgoing

An optional PPSN facility that prevents a particular logical channel from handling incoming calls.

Outgoing Calls Barred

An optional PPSN facility that prevents a DTE from initiating any calls.

Packet

The unit of data switched through a PPSN; normally a user data field accompanied by a header carrying destination and other information.

Packet Assembly/Disassembly (PAD) Facility

A device at a PPSN node that allows access from an asynchronous terminal, such as an LA36. The terminal connects to the PAD and the PAD puts the terminal's input data into packets (assembles) and takes the terminal's output data out of packets (disassembles).

Packet Control

The functions concerned with the correct routing and reception of individual packets through the network.

Packet-mode DTE

A DTE that can handle data in packet form. This implies a capability for assembling and disassembling packets. A computer is one type of packet-mode DTE.

Packet Receive Sequence Number (P(R))

The P(R) number indicates that all packets up to that number minus one have been received. The P(R) number thus authorizes the transmission of further packets by updating the lower window edge.

Packet Send Sequence Number (P(S))

The P(S) number specifies the position of a packet in a sequential stream. The number starts at zero for the first packet and increases by one for each successive packet sent on one logical channel in one direction. The P(S) number can be either modulo 8 or modulo 128 although modulo 8 is the default for all PPSNs. A packet can only be transmitted if its P(S) is greater than or equal to the lower window edge and less than the upper window edge.

Packet Switching

A data transmission process, utilizing addressed packets, whereby a channel is occupied only for the duration of transmission of the packet.

NOTE

In certain data communication networks, the data can be formatted into a packet or divided and then formatted into a number of packets (either by the data terminal equipment or by equipment within the network) for transmission and multiplexing purposes.

Packetnet System Interface (PSI)

The collective name for the hardware and software products that allow various DIGITAL operating systems to participate in a packet switching environment.

Permanent Virtual Circuit (PVC)	A permanent logical association between two DTEs, which is analogous to a leased line. Transmission of packets on a PVC needs no call set up or call clearing by the DTE. Packets are routed directly by the network from one DTE to the other.
Port	A collection of resources that maintain a virtual circuit.
PPSN	See Public Packet Switching Network.
Protocol	An agreed set of rules governing the operation of a communications link.
Public Packet Switching Network (PPSN)	A set of equipment and interconnecting links that provides a packet switching set of equipment and interconnecting links that provides a packet switching communications service to subscribers within a particular country.
PVC	See Permanent Virtual Circuit.
Qualified Data	Data transmitted in a packet in which the Qualifier bit is set. This bit is usually reserved for special applications, such as higher level protocols. For example, X.29 protocol messages are transmitted between PADs as qualified data messages.
Remote DTE	A frame of reference: any DTE in a network other than the one at which the user is located.
Remote Virtual Terminal	A terminal connected to a Packet Assembly/Disassembly (PAD) facility.
Reset	A reset allows a DTE to re-initialize a virtual circuit by resetting the lower window edge and P(S) and P(R) numbers to zero. All Data and Interrupt packets that may be in the network are discarded.
Reverse Charging	An optional PPSN facility that allows a DTE to request that the remote DTE is charged for a particular call.
Start Element	A single 0-bit that marks the start of a character in start-stop transmission.

Start/stop Transmission	Asynchronous transmission in which a group of bits corresponding to a character is preceded by a start element and is followed by a stop element.
Stop Element	Either one or two 1-bits that mark(s) the end of a character in start-stop transmission.
SVC	See Switched Virtual Circuit.
Switched Virtual Circuit (SVC)	A temporary logical association between two DTEs connected to a PPSN which is analogous to connection by a dial-up line. An SVC is set up only when there is data to transmit and is cleared when the data transfer is complete.
Tariff	A published rate for telecommunications services.
Throughput Class Negotiation	An optional PPSN facility that indicates the maximum data rate for a particular virtual circuit. The facility allows a DTE to request a higher or lower data rate depending on the throughput of the packets.
Virtual Circuit	An association between two DTEs connected to a PPSN whereby the two DTEs are able to interact as if a specific circuit is dedicated to them throughout the transmission. In reality, a logical connection is established, and the actual physical circuits are allocated according to route availability, overload conditions, and so on.
Window	The ordered set of consecutive data packets authorized to cross the DTE/DCE interface of the logical channel used for an SVC or PVC in each direction of transmission. The lowest sequence number in the window is called the lower window edge. When an SVC or PVC at the DTE/DCE interface has just been established, the window related to each direction of data transmission has a lower window edge equal to 0. The packet send sequence number of the first data packet not authorized to cross the interface is the value of the upper window edge; that is, the lower window edge plus the window size.
X.3	A CCITT recommendation that specifies the Packet Assembly/Disassembly (PAD) facility in a public data network.

X.25	A CCITT recommendation that specifies the interface between Data Terminal Equipment and Data Circuit-terminating Equipment for equipment operating in the packet mode on public data networks.
X.28	A CCITT recommendation that specifies the DTE/DCE interface for a start-stop mode DTE accessing the Packet Assembly/Disassembly (PAD) facility in a public data network situated in the same country.
X.29	A CCITT recommendation that specifies procedures for the exchange of control information and user data between a packet-mode DTE and a Packet Assembly/Disassembly (PAD) facility.
X.75	A CCITT recommendation that specifies the procedures for communicating between PPSNs.

INDEX

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

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

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____ Telephone _____

Street _____

City _____ State _____ Zip Code or Country _____

------Do Not Tear - Fold Here and Tape------

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MRO1–2/L12
MARLBOROUGH, MA 01752

----- Do Not Tear - Fold Here and Tape------

Cut Along Dotted Line