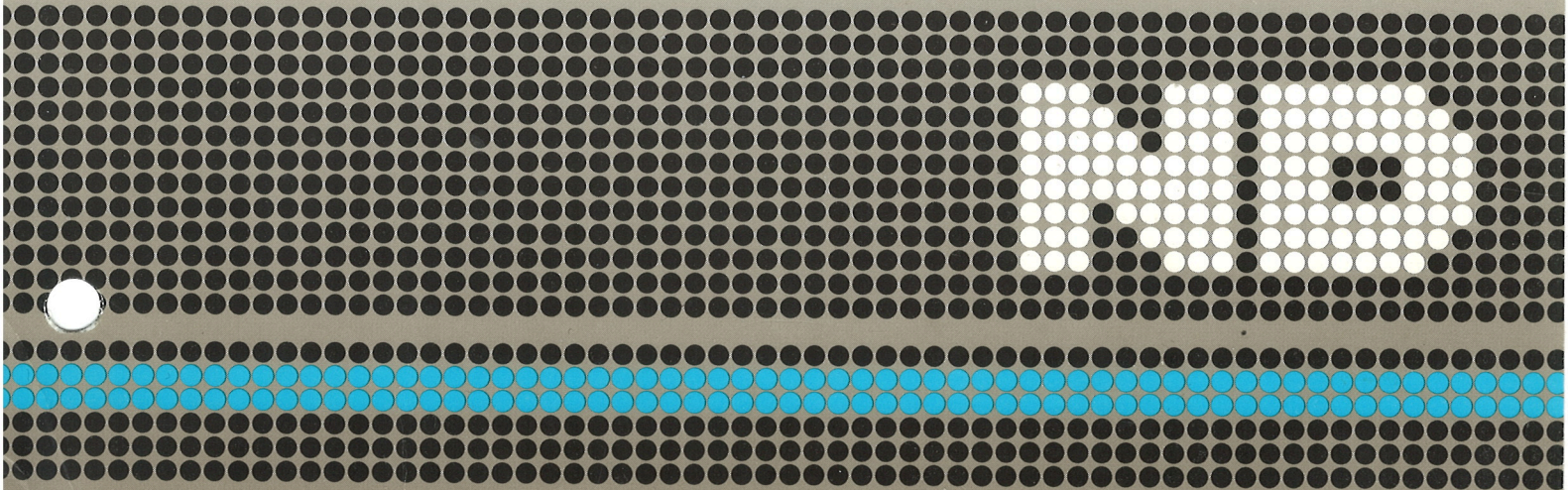


COSMOS X.25 Programmer Guide

ND-60.227.1 EN



COSMOS X.25 Programmer Guide

ND-60.227.1 EN

NOTICE

The information in this document is subject to change without notice. Norsk Data A.S assumes no responsibility for any errors that may appear in this document. Norsk Data A.S assumes no responsibility for the use or reliability of its software on equipment that is not furnished or supported by Norsk Data A.S.

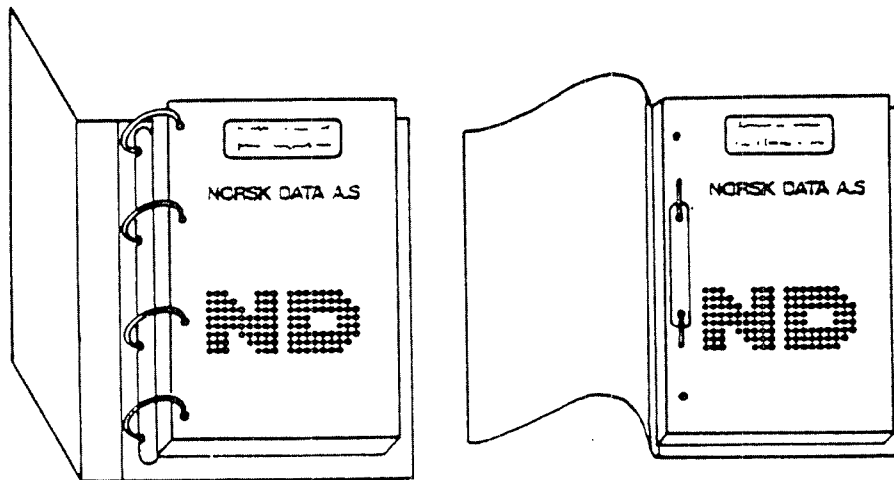
The information described in this document is protected by copyright. It may not be photocopied, reproduced or translated without the prior consent of Norsk Data A.S.

Copyright ©1985 by Norsk Data A.S.

This manual is in loose-leaf form for ease of updating. Old pages may be removed and new pages easily inserted if the manual is revised.

The loose-leaf form also allows you to place the manual in a ring binder (A) for greater protection and convenience of use. Ring binders with 4 rings corresponding to the holes in the manual may be ordered in two widths, 30 mm and 40 mm. Use the order form below.

The manual may also be placed in a plastic cover (B). This cover is more suitable for manuals of less than 100 pages than for large manuals. Plastic covers may also be ordered below.



A: Ring Binder

B: Plastic Cover

Please send your order to the local ND office or (in Norway) to:

Norsk Data A.S

Graphic Center
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

ORDER FORM

I would like to order

..... Ring Binders, 30 mm, at nkr 20,- per binder

..... Ring Binders, 40 mm, at nkr 25,- per binder

..... Plastic Covers at nkr 10,- per cover

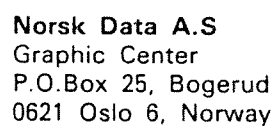
Name

Company

Address

.....

City

COSMOS X.25 Programmer Guide
Publ.No. ND-60.227.1 EN

Manuals can be updated in two ways, new versions and revisions. New versions consist of a complete new manual which replaces the old manual. New versions incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the Customer Support Information (CSI) and can be ordered as described below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and to give an evaluation of the manual. Both detailed and general comments are welcome.

These forms and comments should be sent to:

Documentation Department
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Requests for documentation should be sent to the local ND office or (in Norway) to:

Graphic Center
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Preface:

THE PRODUCT

This manual documents the COSMOS X.25 programmer library.

THE READER

This manual is written for the programmer who needs to write data communication software based on X.25 directly.

PREREQUISITE KNOWLEDGE

The reader must have a general understanding of data communication, knowledge of the SINTRAN III operating system, and programming experience in PLANC. General knowledge about the X.25 recommendation is essential.

RELATED MANUALS

COSMOS Programmer Guide ND-60.164.1
COSMOS X.25 Option Operator Guide ND-30.034.1

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1 INTRODUCTION TO THE X.25 PROGRAMMER LIBRARY	1
1.1 Introduction	3
1.2 Prerequisites for Establishing a Connection	3
1.2.1 Initialization	3
1.2.2 Establishing Communication with the X.25 PL	3
1.2.3 Setting up Connection End Points (CEPs)	4
1.3 Establishing a Connection	5
1.3.1 Requesting an Outgoing Connection	5
1.3.2 Accepting or Rejecting an Incoming Connection	6
1.4 Data Transfer	7
1.4.1 Normal Data	7
1.4.2 Expedited Data	8
1.5 Reset	8
1.5.1 Requesting a Reset	8
1.5.2 Accepting a Reset	8
1.6 Disconnect And Closing Down	8
1.7 The Wait routine and event mechanism	9
1.8 State Diagrams	10
2 X.25/PLANC REFERENCE GUIDE	13
2.1 General Information about the Routines	15
2.2 Summary of the different routines	15
2.3 Record Types	15
2.3.1 Specification of a DTEaddress field	16
2.3.2 Structures for facilities field	16
2.3.3 Quality of service	17
2.3.4 Structure for user data	17
2.3.5 Structure of an Event	17
3 SAMPLE PROGRAMS	43
 <u>APPENDIX</u>	
A ERROR CODES	51
Index	55

CHAPTER 1

INTRODUCTION TO THE X.25 PROGRAMMER LIBRARY

1 INTRODUCTION TO THE X.25 PROGRAMMER LIBRARY

1.1 Introduction

This manual describes the COSMOS X.25 programmer library, which enables programs to set up connections and transfer data across an X.25 packet switching network. The library converts the caller's requests into messages, sent via XMSG, to an X.25 Packet Level (PL) module. It also converts messages received from the PL into events reported to the caller via a wait mechanism. The X.25 PL is described in the COSMOS X.25 Option Operator Guide, chapter 1.

The current implementation of the COSMOS X.25 programmer library supports a PLANC interface only. Some parameters are included in the interface but are not used by the current implementation of the library. These parameters are included to allow for future enhancements.

1.2 Prerequisites for Establishing a Connection

1.2.1 Initialization

The first routine to be called must be the initialization routine X25PINI. The purpose is to define a work area in your program so the library can work in a reentrant way. The library will use the work area for some of its own internal data structures. The library returns a workid, which is used to identify the work area in subsequent calls.

1.2.2 Establishing Communication with the X.25 PL

A communication channel between your program and a particular X.25 PL module has to be created. This is done by calling the routine X25POPNI. The communication channel is called a port and identified by a portid assigned to you by the library. The X.25 PL module controls one (and only one) physical communication line at a time. If you need to set up X.25 connections over more than one physical line, you must repeat the X25POPNI call to establish communication with the same number of X.25 PL modules.

A Network Service Access Point (NSAP or just SAP) defines a conceptual point which provides access to the X.25 network service. The library returns an identifier (portid) belonging to the SAP. A SAP is implemented as a DTE number plus a SAP suffix. For the different types of SAP suffixes, please see the description of the X25PATR routine.

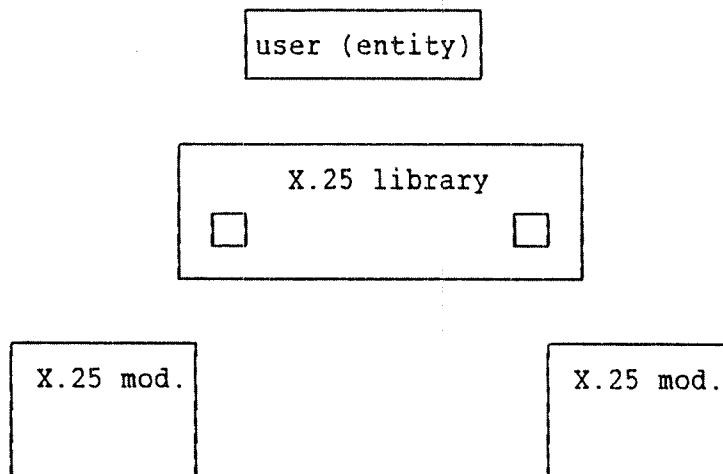


Fig. 1. Establishing 2 channels to the X25 PL

1.2.3 Setting up Connection End Points (CEPs)

A Connection End Point (CEP) defines a conceptual point at which a connection can take place. CEPs are used by the library and the PL to allocate certain resources for future connections. If an established connection is subsequently disconnected, these resources are not released, but are held ready for the next connection on the same CEP. The resources are only released when the CEP is deleted.

The number of CEPs needed is the sum of:

- a) The maximum number of simultaneous Switched Virtual Circuit (SVC) connections.
- b) The number of Permanent Virtual Circuits (PVCs) to be used.

A CEP is defined by calling the Attach Request routine X25PATR. The library and PL allocate resources such as internal buffers, and then take action depending on the type of connection required (incoming SVC, outgoing SVC, or PVC):

- 1) Incoming SVC. The PL informs the library (and hence the user program) of incoming calls that satisfy the required conditions. That means either the incoming call specifies the DTE subaddress corresponding to the CEP, or the incoming call data field (CUD) contains the string required by the CEP. Incoming calls that do not satisfy these conditions, are not routed to this CEP.

- 2) Outgoing SVC. No special actions are taken.
- 3) PVC. The PL reserves the specified PVC number for exclusive use by this CEP.

The library returns an identifier (the cepid) to be used in subsequent calls. The user also specifies an identifier (ucepid) when calling X25PATR. The library uses this ucepid when informing the user that an event has occurred. This mechanism enables both the library and the user program to use simple look-up tables to identify information about CEPs.

When the X25PATR routine has been called, you should wait for an Attach Confirmation event. Changes to the NIDU sizes or other values made by the PL can be examined by calling the routine X25PATC.

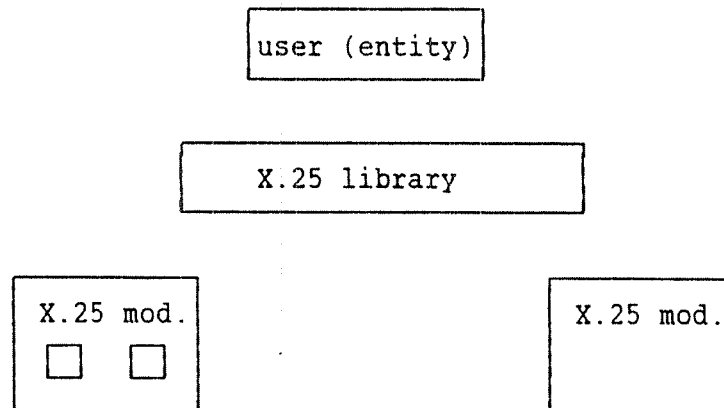


Fig. 2. Establishing 2 CEPs is done by calling X25PATR twice

1.3 Establishing a Connection

1.3.1 Requesting an Outgoing Connection

You initiate an outgoing connection by calling the Connect Request routine X25PCNR. This specifies the DTE number of the called CEP, any special facilities required (for example reversed charging, fast select option), and any user data to be sent with the call. You must then wait for an event to arrive from the other side. This will be either:

- a) A connection confirmation event, which tells you that the call has been accepted. The routine X25PCNC should be called when this event occurs in order to obtain the DTE address of the responder (may be different from the called address because of re-routing), the facilities granted by the other

side, and any user data sent with the connection acceptance. The connection is now established and data transfer can take place.

- b) A disconnect indication event, which tells you that the call has been refused. The reason for refusal is given in the event data. Any user data sent with the disconnect indication can be obtained by calling the routine X25PDCI.

1.3.2 Accepting or Rejecting an Incoming Connection

An incoming connection satisfying the conditions for a CEP is signalled by a connection indication event. The called and calling DTE addresses, requested facilities and user data may be obtained by calling the routine X25PCNI. Note that the facilities may have been reduced by the network from those requested by the other side.

If the CEP was set up specifying a 4-step connection procedure, you must decide whether to accept or reject the connection request. If you decide to accept it, you call the routine X25PCNA. This specifies the full DTE address of the responding CEP, the facilities granted, and any user data to be sent with the connection acceptance. The connection is now established, and data transfer can take place. If you decide to reject the connection request, you call the routine X25PDCR, specifying the reason for refusal.

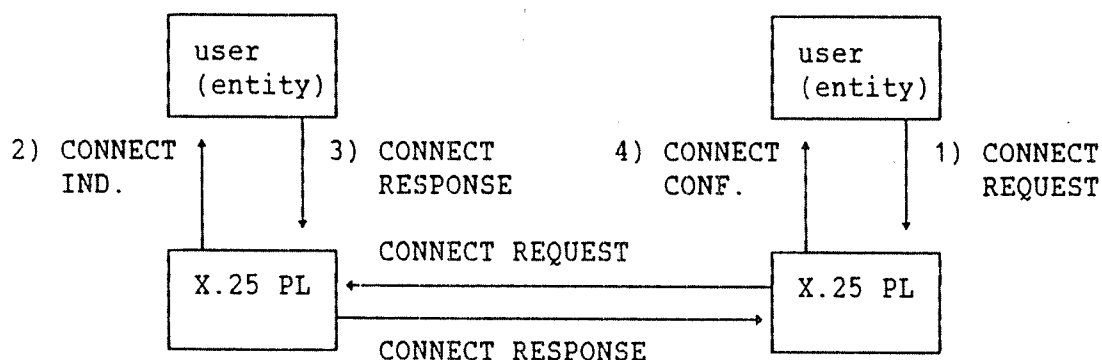


Fig. 3. 4-step connection procedure

If the CEP was set up specifying a 3-step connection procedure, the X25PCNA call should not be made. This is because this procedure assumes that the connection will always be accepted, so a connection acceptance is sent automatically by the PL. In this case, the connection is established when the connection indication event is received.

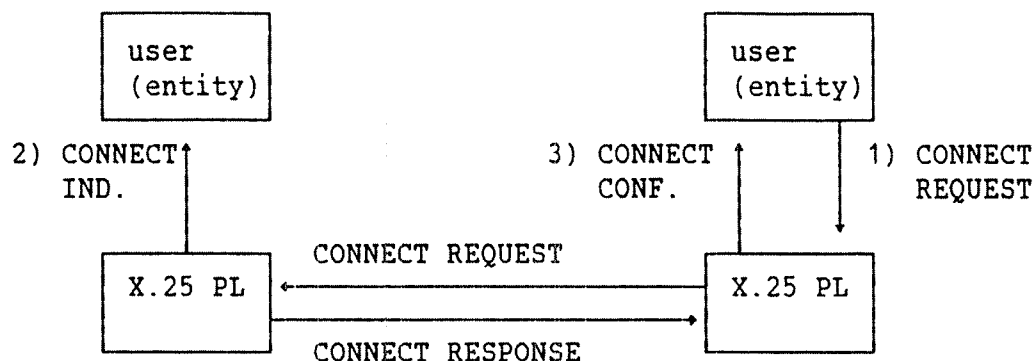


Fig. 4. 3-step connection procedure

1.4 Data Transfer

1.4.1 Normal Data

Once a connection is established, normal data is sent to the remote user in units called NSDUs and NIDUs.

A Network Service Data Unit (NSDU) is a logical unit of data transported between network layer entities. An NSDU is subdivided into Network Interface Data Units (NIDUs). The Data Request routine X25PDAR transmits one NIDU, and these may be grouped into an NSDU by specifying the Moredata flag with all NIDUs except the last. The Qualifier flag may also be specified to distinguish between two types of data (for example commands and ordinary data). Note that the same value of the Qualifier flag must be specified for all NIDUs comprising a single NSDU.

That data has arrived from the other end is specified by the Data Indication event. You may obtain the data by calling the routine X25PDAI, specifying a buffer where you want the data to be stored. The flag values specified by the remote end are returned.

An alternative method of obtaining the data, and one that improves the efficiency of the library and X.25 PL, is to call the routine X25PPRB before waiting for a data indication event. X25PPRB specifies a buffer in advance of the event, so the data can be transferred without delay. When the data indication event occurs, X25PDAI can be called, and the details of the buffer used are returned. Note that X25PPRB can only be used for normal data, and only one buffer can be specified per incoming NIDU.

1.4.2 Expedited Data

This is a limited amount of data (currently one byte at a time) which you can send to the remote user, during the data transfer phase, outside the ordinary data stream. Expedited data is sent by calling X25PEDR. The arrival of the expedited data will be signalled to the remote user, by the Expedited Data indication event, before any normal data subsequently sent. The remote user obtains the expedited data by calling the routine X25PEDI.

Only one item of expedited data can be sent at one time. You have to wait until you receive an Expedited Data confirmation event before you send more expedited data.

1.5 Reset

1.5.1 Requesting a Reset

You request a reset by calling the routine X25PRSR. This causes all outstanding events to be lost, and may cause the loss of data previously sent. You must then wait for a Reset Confirmation event before sending any more data across the connection.

1.5.2 Accepting a Reset

When a Reset Indication event occurs, any other outstanding events are lost. You must reply by calling the Reset Accept routine X25PRSA.

1.6 Disconnect And Closing Down

An established connection is disconnected by calling the routine X25PDCR. This results in a Disconnect Indication event being received by the other side. No more data can be sent on this connection. However, a new connection cannot be set up on the same CEP before a Disconnect Confirmation event is received from the PL.

A CEP can be removed by calling the routine X25PDTR. The cepid becomes invalid and must not be used in subsequent calls.

Communication with an X.25 PL module can be closed down by calling the routine X25PCLS. The portid becomes invalid and must not be used in subsequent calls.

1.7 The Wait routine and event mechanism

In order to wait for incoming events, the user program must call the routine X25PWA1. The parameters that can be specified are:

Timeout: The maximum length of time you have decided to wait. A Timeout event will occur if this time expires. The timeout may be of infinite duration.

Requested Event: Specifies the event(s) you wait for and which CEPs they should occur on. You specify the events as a bit mask (see values in the table of events) or anyevent may be specified. Events on a single CEP, all CEPs on a port, or all CEPs may be specified.

When a suitable event occurs, the X25PWA1 routine returns the control to you, and indicates which type of event has occurred on which CEP. Only one event is signalled by each call of X25PWA1. Any other events that occur are stored by the library, and can be picked up by subsequent calls of X25PWA1.

The symbols for the event codes are defined in the X25P:DEFS file. You may specify a whole set of events by using logical OR to form a bit mask.

Event code	Value	Explanation
X25evtime	1	Timeout
X25evatcf	2	Attach confirmation
X25evcnin	4	Connect indication
X25evcncf	8	Connect confirmation
X25evdain	16	Data indication
X25evxdin	64	Expedited data indication
X25evxdcf	128	Expedited data confirmation
X25evrsin	256	Reset indication
X25evrscf	512	Reset confirmation
X25evdcin	1024	Disconnect indication
X25evdccf	2048	Disconnect confirmation
X25everin	4096	Error indication
X25evunkn	8192	Unrecognized XMSG interrupt
X25evothr	16384	Other interrupt
X25evcrdt	~32768	May send more data

The reason for X25evunkn is that you may combine calls to the X.25 library and to XMSG in the same RT program. As you performed the X25PWA1 call, to wait for an event, a message may arrive on a port opened by an XMSG call. The X25evunkn tells you that this has happened.

The X25PWA1 routine performs the 'tmout' monitor call internally. If your RT-program becomes rescheduled for execution, by for example another program, the X25evothr event will occur.

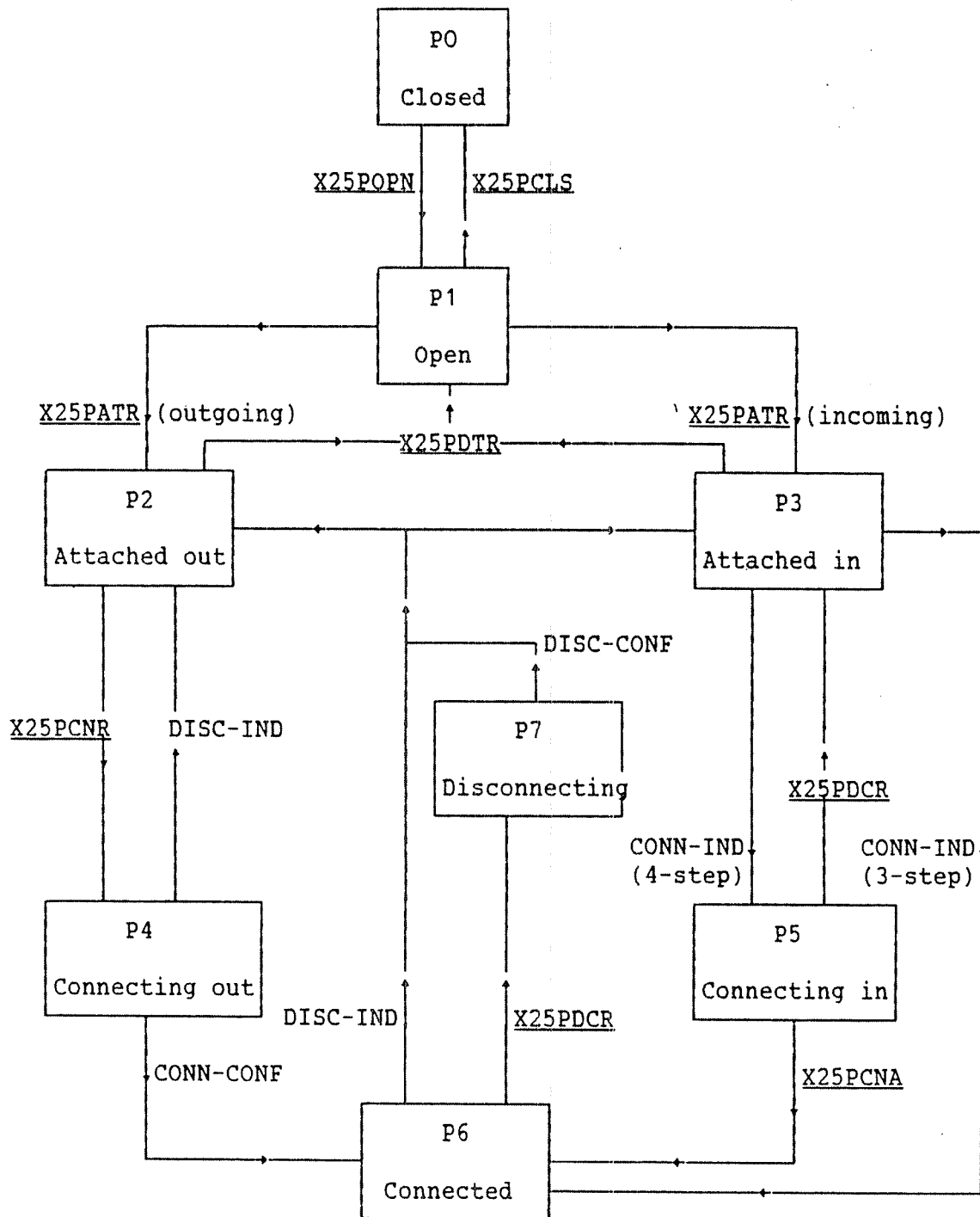
1.8 State DiagramsFig. 5. Setup and release of an association

Fig. 5 describes the state transitions during the setup and release of the associations between two CEPs as it is seen from the user. X25PDTR can be sent in any state P2 to P7; transition is to P1. Underlined actions indicate procedure calls, other actions indicate events.

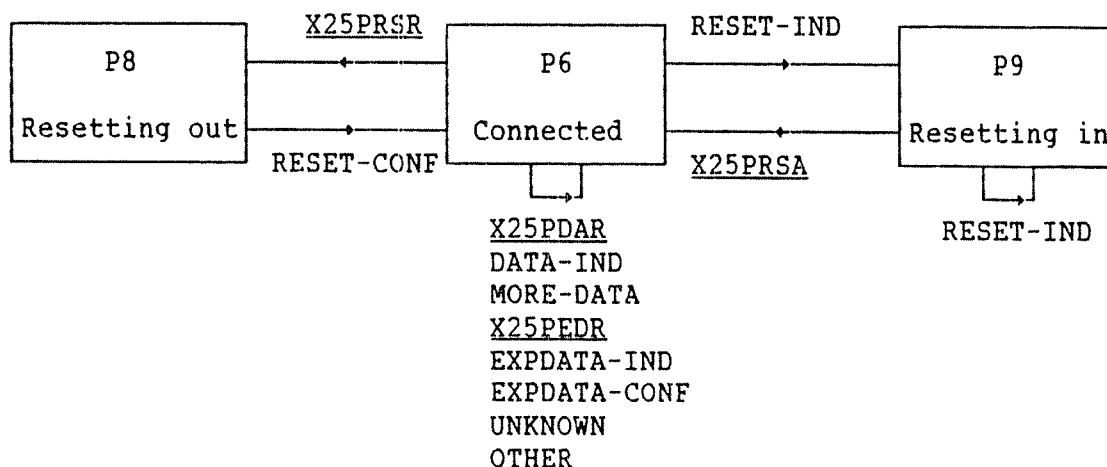


Fig. 6. Data transfer

Fig. 6 shows the possible transitions in the data phase.

C H A P T E R 2

X.25/PLANC REFERENCE GUIDE

2 X.25/PLANC REFERENCE GUIDE

2.1 General Information about the Routines

At present, the routine implementation is in PLANC only. The implementation is as follows: ROUTINE VOID,VOID (parameters.....)
Example of call: X25PEDR(plcepid,xdata,retstat)

Normal return status is zero. If you include the file X25P:DEFS in your source code, you may use the symbol X25ok for the zero status. A list of the error codes plus their corresponding symbols is provided in appendix A.

2.2 Summary of the different routines

Routine	Purpose
X25PATC	Attach confirmation, response to an X25PATR
X25PATR	Attach request, defines a connection end point (CEP)
X25PCLS	Close port, closes connection with an X.25 module
X25PCNA	Connection response
X25PCNC	Connection confirmation
X25PCNI	Connection indication
X25PCNR	Connection request
X25PDAI	Data indication, data is coming in
X25PDAR	Data request, data is going out
X25PDCI	Disconnect indication
X25PDCR	Disconnect request
X25PDTR	Detach request, deletes a CEP
X25PEDI	Expedited data indication
X25PEDR	Expedited data request
X25PINI	Initialization, must be the first call
X25POPNI	Open port request, establishes communication with PL
X25PPRB	Provide buffer for incoming NIDU
X25PRSA	Reset response
X25PRSR	Reset request
X25PWAI	Wait for event to occur

2.3 Record Types

All the records are defined in the X25P:DEFS file. You should include it in your program.

2.3.1 Specification of a DTEaddress field

```

TYPE X25DTEtype = RECORD
    BYTES: X25DTEmainaddress(0:X25maxDTEmainsize-1)
    BYTES: X25DTEsubaddress(0:X25maxDTEsubsize-1)
ENDRECORD

```

X25maxDTEmainsize = 16 and X25maxDTEsubsize = 4.

2.3.2 Structures for facilities field

```

TYPE X25Psize = ENUMERATION(X25badps1,X25badps2,X25badps4,X25badps8,&
    X25ps16,X25ps32,X25ps64,X25ps128,X25ps256,&
    X25ps512,X25ps1024)

```

Note that the values X25ps16, X25ps32 etc. correspond to packet sizes of 16 bytes, 32 bytes etc. The values X25badps1 to X25badps8 are dummy and should not be specified.

```

TYPE X25facilities = RECORD
    BOOLEAN: X25Revcharge
    INTEGER: X25Cug
    BOOLEAN: X25Dbituse
    INTEGER: X25SendWsize
    INTEGER: X25RcveWsize
    X25Psize: X25SendPsize
    X25Psize: X25RcvePsize
    INTEGER: X25SendTclass
    INTEGER: X25RcveTclass
    INTEGER: X25FastSelect
ENDRECORD

```

The following default values may be used:

X25defpsize : Default packet size is 128 bytes.

X25defcug : No closed user group is specified.

X25defwsize : Default window size is 2.

X25deftclass : Default throughput class is 9600.

X25deffastselect : No fast select.

The following fast select values may be used:

X25fsnormal : Full fast select.

X25fsrestricted : Restriction on response.

2.3.3 Quality of service

Currently, only default quality of service is provided:

```
TYPE X25qs = RECORD
    BOOLEAN: X25qsisdefault
ENDRECORD
```

X25qsisdefault is always set to TRUE.

2.3.4 Structure for user data

```
TYPE X25Userdata = RECORD
    X25Intaddress: X25Udaddress
    INTEGER: X25Udlength
ENDRECORD
```

X25intaddress is Integer4 in MC68000 (PIOC), otherwise it is Integer.

2.3.5 Structure of an Event

An event has the following structure:

```
TYPE X25ev = RECORD
    INTEGER: X25evid
    INTEGER: X25evcode
ENDRECORD
```

X25evid may be a plcepid (specifying a particular cep), a portid (specifying any cep in this port), or the workid (specifying any cep on any port). X25evcode specifies the events to be waited for, or the single event you received.

Routine name : <i>X25PATC</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	retstat Integer	W	Return status.
3	sizes X25nidusizes P	W	Resources given by the local PL.
4	queueresources X25qresources P	W	May be used in a future version. Should be set to NIL at present.

FUNCTION. . : Attach confirmation.

Explanation : The X25patc is used to get the response to an X25patr. Even if the request is granted, the PL may have reduced the sizes. At present, queueresources is always set to NIL.

The sizes parameter contains the following values:

Maximum NIDU size for input.
Maximum NIDU size for output.
Minimum buffer pool request (not used at present).
Maximum buffer pool request (not used at present).

The maximum NIDU sizes are used to determine the size of the XMSG buffers. If sizes is NIL, both NIDU sizes are set to 128. The type X25nidusizes is defined as follows:

```
TYPE X25NIDUsizes = RECORD
    INTEGER: X25MaxinNIDUsize
    INTEGER: X25MaxoutNIDUsize
    INTEGER: X25Minbufferpool
    INTEGER: X25Maxbufferpool
ENDRECORD
```

USAGE . . . : X25Patc(Plcepid,Retstat,Sizes,Queueresources)

Routine name : <i>X25PATR</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	portid Integer	R	The value returned by X25POPN.
2	ucepid Integer	R	Your identification of the CEP.
3	conndetails X25conndetails	R	The characteristics of a connection. See below.
4	prealloc Boolean	R	Datafield pre-allocation flag. Not used at present.
5	sizes X25nidusizes P	R	Resources required from the local PL. If specified as NIL, defaults are used.
6	queueresources X25qresources P	R	Should be set to NIL at present. May be used in a future version.
7	retstat Integer	W	Return status.
8	plcepid Integer	W	CEP identification used by X25.

FUNCTION. . : Attach request.

Explanation : This call is used to define a connection endpoint (CEP), and the PL will allocate the necessary resources such as datafields. X25patr is necessary for incoming as well as outgoing connections.

Conndetails describes the type of connection. It can take one of the following values: PVC, SVC outgoing, or SVC incoming.

For PVCs the PVC number is also required.

For SVCs information about connection protocol and SAP suffix is required.

Connection protocol defines the behavior of the PL for incoming calls. Possible values are 3-step connect or 4-step connect. Please refer to page 6.

SAP suffix is used to select conditions for receiving incoming calls, either by selecting a DTE subaddress or by matching some or all of the contents of the Call User Data (CUD) field (in the packet header). For outgoing calls, only the DTE subaddress may be specified, and this will be appended to the calling address when a Connect Request is made.

The SAP suffix consists of either a DTE subaddress (max. 4 digits), a CUD string specification, or a CUD byte specification. If you set the SAP suffix to NIL, then the DTE subaddress is selected.

CUDstring specifies an ASCII character string (max. length 16) and a start position. Incoming calls are routed to this CEP if the string appears in the corresponding position in the incoming CUD.

CUDbyte is similar to CUDstring except that the values are specified as integers rather than bytes. This allows special values to be used, e.g. to accept any value for a byte (wild card).

The structures for specifying the connection details are defined as follows:

```
TYPE X25Connections = ENUMERATION(X25PVC,&
                                   X25SVCincoming,X25SVCoutgoing)
```

```
TYPE X25ConnDetails = RECORD
    X25connections: X25Conntype
ENDRECORD
```

```
TYPE X25PVCconnection = X25ConnDetails&
RECORD
    INTEGER: X25PVCNumber
ENDRECORD
```

```
TYPE X25inSVCconnection = X25ConnDetails&
RECORD
    X25SAPsuftype POINTER: X25SAPsuffix
    ENUMERATION(X25threestep,&
                X25fourstep): X25connprotocol
ENDRECORD
```

```
TYPE X25outSVCconnection = X25ConnDetails&
RECORD
    X25DTEsuftype POINTER: X25DTEsuffix
ENDRECORD
```


These are the structures for specifying the SAP suffix:

```
TYPE X25SAPsuftype = RECORD
    ENUMERATION(X25CUDstring,X25CUDbyte,
                &X25DTEsub): X25SAPtype
    INTEGER: X25suflength
ENDRECORD

TYPE X25DTEsuftype = X25SAPsuftype&
RECORD
    BYTES: X25DTEsubaddress(0:X25maxDTEsubsize-1)
ENDRECORD

TYPE X25CUDsuftype = X25SAPsuftype&
RECORD
    INTEGER: X25sufstartpos
ENDRECORD

TYPE X25CUDsufbyte = X25CUDsuftype&
RECORD
    INTEGER ARRAY: X25bytevalues(0:X25maxsufsize-1)
ENDRECORD

TYPE X25CUDsufstring = X25CUDsuftype&
RECORD
    BYTES: X25sufstringvalue(0:X25maxsufsize-1)
ENDRECORD
```

X25maxDTEsubsize = 4 and X25maxsufsize = 16.

A wild card value for X25CUDbyte is: X25anybyte = 1

The sizes parameter contains the following values:

- Maximum NIDU size for input.
- Maximum NIDU size for output.
- Minimum buffer pool request (not used at present).
- Maximum buffer pool request (not used at present).

The maximum NIDU sizes are used to determine the size of the XMSG buffers. If sizes is NIL, both NIDU sizes are set to 128.

The type X25nidusizes is defined as follows:

```
TYPE X25NIDUsizes = RECORD
    INTEGER: X25MaxinNIDUsize
    INTEGER: X25MaxoutNIDUsize
    INTEGER: X25Minbufferpool
    INTEGER: X25Maxbufferpool
ENDRECORD
```

USAGE . . . : X25Patr(Portid,Ucepid,Conndetails,Prealloc,Sizes,&
Queueresources,Retstat,Plcepid)

Routine name : <i>X25PCLS</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	portid Integer	R	Identifier for that part of the work area which this X25 module uses.
2	retstat Integer	W	Return status.

FUNCTION. . . : Close port.

Explanation : By using this call you tell the library that you have finished with an X.25 Packet Level module. The library will check if any CEPs are in use and send detach requests for these. Then it will break the communication with the PL (i.e. close the XMSG port used by the library).

USAGE . . . : X25Pcls(Portid,Retstat)

Routine name : X25PCNA			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	calledDTENumber X25dtetype	R	The DTE number of this end.
3	facilities X25facilities P	R	Facilities to be encoded into the CALL ACCEPTED facilities field.
4	qos X25qs P	R	Quality of service.
5	udata X25userdata	R	User data to be sent with the connection response.
6	retstat Integer	W	Return status.

FUNCTION. . . : Connection response.

Explanation : This call is used to accept a connection request. The calledDTEnum can be used to indicate the actual DTE number and subaddress used. The facilities specified in the connection response may be reduced from those given by the connection indication. User data is only permitted if the facilities include fast select.

USAGE . . . : X25Pcna(Plcepid,CalledDTENumber,Facilities,Qos,& Udata,Retstat)

Routine name : <i>X25PCNC</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	retstat Integer	W	Return status.
3	calledDTENumber X25dtetype	W	The DTE number of the remote end.
4	facilities X25facilities P	W	Facilities specified by the connection response.
5	qos X25qs P	W	Quality of service.
6	udata X25userdata	W	User data.

FUNCTION. . . : Connection confirmation.

Explanation : A connection response coming from the remote CEP will be perceived by your local CEP as a connection confirmation. This is flagged as an X25evcnf event. The W parameters in X25pcnc are the data associated with this event.

USAGE . . . : X25Pcnc(Plcepid,Retstat,CalledDTENumber,&
Facilities,Qos,Udata)

Routine name : X25PCNI			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	retstat Integer	W	Return status.
3	calledDTENumber X25dtetype	W	The local DTE number.
4	callingDTENumber X25dtetype	W	The DTE number of the remote end.
5	facilities X25facilities P	W	Facilities set in the X25PCNR call may have been reduced by PL or network.
6	qos X25qs P	W	Quality of service.
7	udata X25userdata	W	User data sent with the connect request.

FUNCTION. . . : Connection indication.

Explanation : When a remote CEP sends a connection request to you, then your packet level will perceive it as a connection indication. This will be signalled to you as an X25evcnin event. The W parameters in X25PCNI are information associated with this event.

USAGE . . . : X25Pcni(Plcepid,Retstat,CalledDTENumber,&CallingDTENumber,Facilities,Qos,Udata)

Routine name : X25PCNR			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	calledDTENumber X25dtetype	R	The DTE number of the remote end.
3	facilities X25facilities P	R	Facilities to be encoded into the CALL REQUEST facilities field.
4	qos X25qs P	R	Quality of service.
5	udata X25userdata	R	User data to be sent with the connect request.
6	retstat Integer	W	Return status.

FUNCTION. . : Connection request.

Explanation : A connection between the local and remote CEPs is initiated. Before the exchange of NSDUs can take place, the connection must be completely established.

USAGE . . . : X25cncr(plcepid,calledDTENumber,facilities,qos,udata,&retstat)

Routine name : <i>X25PDAI</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	retstat Integer	W	Return status.
3	flags Integer	W	M-bit and Q-bit. Please see below.
4	udata X25userdata	RW	Data coming from the remote CEP.
5	ubufid Integer	RW	The buffer identifier, in case buffer was provided by X25PPRB.

FUNCTION. . . : Data indication.

Explanation : When data is received, it is signalled to you via the X25evdain event. X25PDAI is called to obtain one NIDU. If you have not provided the packet level with a buffer by using X25PPRB, the buffer specified by the parameters udata and ubufid is used. Otherwise, the buffer details specified in a previous call to X25PPRB are returned in the parameters udata and ubufid.

The flags parameter is set to the value specified in the X25PDAR sent by the remote end.

USAGE . . . : X25Pdai(Plcepid,Retstat,Flags,Udata,Ubufid)

Routine name : X25PDAR			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	flags Integer	R	M-bit and Q-bit. Please see below.
3	udata X25userdata	R	Data to be sent to the remote CEP.
4	retstat Integer	W	Return status.

FUNCTION. . : Data request.

Explanation : This call is used to send one NIDU to the other end of the connection. Here are the possible values of retstat for this call:

X25ok The data is sent to the local PL.
 X25ercrdt Retry later. Data cannot at present be sent to the PL since the window is full. The caller must wait for an X25evcrdt event before retrying X25PDAR.

The M-bit (more data) is defined by setting flags to X25moredata. The Q-bit (qualifier) is defined by setting flags to X25qualifier. Note that the Q-bit must be the same value in all NIDUs "linked" by the use of the M-bit. The combination X25moredata + X25qualifier may be used.

The symbols are included in the X25P:DEFS file.

USAGE . . . : X25Pdar(Plcepid,Flags,Udata,Retstat)

Routine name : <i>X25PDCI</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	retstat Integer	W	Return status.
3	originator <i>X25originator</i> <i>X25OrigType</i>	W	The disconnect may be user originated or provider originated.
4	discreason Integer	W	Reason for disconnect.
5	udata X25userdata	W	User data is only permitted with fast select.

FUNCTION. . : Disconnect indication.

Explanation : An incoming disconnect request is perceived by your packet level as a disconnect indication. This is signalled to you via an X25evdcin event. X25PDCI gives you the information associated with this event.

Originator is specified by the following record defined in the X25P:DEFS file:

```
TYPE X25origtype = RECORD
    X25originator : X25orig
ENDRECORD
```

```
TYPE X25originator = ENUMERATION (X25user,X25provider).
```

USAGE . . . : X25Pdci(Plcepid,Retstat,Originator,Discreason,Udata)

Routine name : <i>X25PDCR</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	discreason Integer	R	Reason for disconnect.
3	udata X25userdata	R	User data. Only permitted with fast select. PL does not guarantee delivery.
4	retstat Integer	W	Return status.

FUNCTION. . . : Disconnect request.

Explanation : The purpose of this call is one of the following:

- 1) To refuse a connection request from a remote user, signalled by an X25evcnin event.
- 2) To terminate an established connection.

Any outstanding buffered data not yet sent to the remote CEP is flushed. The same is true for received data not yet given to you.

When the disconnect is complete, this will be signalled to you via the X25evdccf event.

No other calls may be invoked on the connection after you have done an X25pdcrc.

USAGE . . . : X25Pdcrc(Plcepid,Discreason,Udata,Retstat)

Routine name : <i>X25PDTR</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	retstat Integer	W	Return status.

FUNCTION. . . : Detach request.

Explanation : This call is used to release the internal resources used by a CEP, and delete the CEP.

USAGE . . . : X25Pdtr(plcepid,retstat)

Routine name : <i>X25PEDI</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	retstat Integer	W	Return status.
3	xdata X25userdata	W	Buffer with expedited data. Ignore any byte after the first.

FUNCTION. . . : Expedited data indication.

Explanation : You receive the expedited data that was signalled to you, via the X25evxdin event, by using this call.

Only one byte of expedited data is sent.

USAGE . . . : X25Pedi(Plcepid,Retstat,Xdata)

Routine name : <i>X25PEDR</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	xdata X25userdata	R	Buffer with expedited data. Any byte after the first is ignored at present.
3	retstat Integer	W	Return status.

FUNCTION. . . : Expedited data request.

Explanation : You can send user data with high priority to the other end by using this call. Only one X25PEDR call can be given at a time. The completion of a previous X25PEDR is signalled via the X25evxdcf event. After this you may call the routine again.

At present only one byte of expedited data can be sent in each request.

USAGE . . . : X25Pedr(Plcepid,Xdata,Retstat)

Routine name : <i>X25PINI</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	workarea Integer Array	R	Work area.
2	maxvalues X25maxima	R	Determine the size of the work area needed.
3	retstat Integer	W	Return status.
4	workid Integer	W	Identifier specifying this work area in subsequent calls.

FUNCTION. . : Initialization. This must be the first call to the X.25 library.

Explanation : The purpose of this call is to define a work area in your data space for internal data structures used by the X.25 Library. This allows the library to be re-entrant (i.e. only one copy is required for all users).

The work area contains information about X.25 ports, Connection End Points (CEPs) and user data buffer descriptors. The X.25 library is responsible for the operations on the work area, and will format it appropriately. If the area is not large enough for the specified request, an error will be returned.

maxvalues is specified by the X25Maxima record defined in the X25P:DEFS file:

```

TYPE X25Maxima = RECORD
  INTEGER: X25ports
  INTEGER: X25ceps
  INTEGER: X25conns
  INTEGER: X25buffers
ENDRECORD

```

X25conns is the maximum number of simultaneous connections. X25buffers is the maximum number of provided buffers. If you do not set these, the library sets them both to the value of X25ceps.

X25ports is the maximum number of ports. X25ceps is the maximum number of CEPs.

USAGE . . . : X25Pini(workarea,maxvalues,retstat,workid)

Routine name : <i>X25POP</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	workid Integer	R	Workid given to you by X25PINI.
2	X25systemid Bytes	R	COSMOS system containing the PL. Default is same system as for calling program.
3	X25moduleno Integer	R	Number of the X.25 module to be accessed.
4	optype X25optype	R	Type of operation. At present only non-queued operation may be used.
5	entityid Integer	R	Identifies the requesting entity. Default is X25defentity = -1.
6	retstat Integer	W	Return status.
7	portid Integer	W	Identifier for that part of the work area which this X25 module uses.

FUNCTION. . : Open port request.

Explanation : Communication with a local X.25 Packet Level module is established by performing this call.

X25systemid and X25moduleno are used to connect to a local X.25 PL. If it is required to connect with more than one X.25 module (e.g. if there are several physical communication lines), an X25POP call is needed for each X.25 module.

One entity may consist of several processes (RT-programs), but in most cases only one process constitutes the entity. Single-process entities will use the default value in this field (the entity id will be set by the library to the RT-program address). Multi-process entities will have to predefine an identifier, and this must not conflict with other multi-process entity identifiers.

optype is defined by:

TYPE X25Optype = ENUMERATION(Queued,Nonqueued)

So far only Nonqueued may be used.

USAGE . . . : X25Popn(Workid,X25systemid,X25moduleno,Optype,&
Entityid,Retstat,Portid)

Routine name : X25PPRB			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	udata X25userdata	R	Data coming from the remote CEP.
3	ubufid Integer	R	Buffer identifier. Used for retrieving the buffer in X25PDAI.
4	retstat Integer	W	Return status.

FUNCTION. . . : Provide buffer.

Explanation : Incoming data can only be passed to the user when a buffer is provided for it. In the simple case, this buffer is specified by the X25PDAI routine which is called after the X25evdain event has occurred. However, this means that the data must be kept in an XMSG buffer until the routine is called. This can be inefficient and clog XMSG.

To avoid the problem, it is possible to specify user data buffers before X25PWAI is called. This is done with the X25PPRB call. When data arrives, it is immediately put into a provided buffer of sufficient size. In the subsequent call of the X25PDAI routine, the specified value of the user data parameter is ignored, but the address, length and buffer identifier of the buffer used will be returned.

At present only one specified buffer is allowed per incoming NIDU. If the buffer is not big enough to hold the data, as much of the data as possible is put in the buffer. The rest is lost, and a warning value (X25erbfisz) is returned in retstat.

Note that X25PPRB only applies to normal data, and not to user data passed by connect or disconnect calls, nor to expedited data.

USAGE . . . : X25Pprb(plcepid,udata,udbufid,retstat)

Routine name : <i>X25PRSA</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	retstat Integer	W	Return status.

FUNCTION. . . : Reset response.

Explanation : The reset operation is a confirmed service. You must send a reset response to tell the remote CEP that you have received the reset indication event. The remote CEP is informed by the X25evrscf event that the reset request was confirmed.

USAGE . . . : X25Prsa(plcepid,retstat)

Routine name : <i>X25PRSR</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	plcepid Integer	R	CEP identification used by X25.
2	resetreason Integer	R	Reason for the reset.
3	retstat Integer	W	Return status.

FUNCTION. . . : Reset request.

Explanation : The X25PRSR call clears the pipe between the two CEPs. Unacknowledged user data and expedited user data will remain unacknowledged.

The reason for reset is implemented as follows:

High order byte - Reset cause.
Low order byte - Diagnostic code.

The remote CEP perceives the reset request as a reset indication. The remote user is informed by the X25evrsin event. There is no corresponding reset indication call, since all information needed is supplied in the event data.

USAGE . . . : X25Prsr(Plcepid,Resetreason,Retstat)

Routine name : <i>X25PWA1</i>			
No:	Parameter Name/ Type:	R/W	Explanation:
1	timeout X25tm	R	Maximum waiting time for event.
2	regevent X25ev	R	Requested event.
3	retstat Integer	W	Return status.
4	actevent X25ev	W	Actual event.
5	acteventdata Integer array	W	Event data for actual event.

FUNCTION. . : Wait for event to occur.

Explanation : The X25tm type in timeout has the following definition:

```

TYPE X25tm = RECORD
    X25tmcode: X25tmunits
    INTEGER: X25tmlength
ENDRECORD

```

X25tmcode is defined as follows: TYPE X25tmcode =
ENUMERATION(X25tmbasic,X25tmsecs,X25tmmins,X25tmhrs)

These units denote: basic time units, seconds, minutes,
and hours respectively.

X25tmlength describes the number of time units.

A value of X25infinite for <timeout.x25tmlength>
specifies an infinite timeout period.

regevent specifies the event to wait for. It consists of
an id and an eventcode. The id may be a plcepid
(specifying a particular cep), a portid (specifying any
cep in this port), or the workid (specifying any cep on
any port). The eventcode specifies the events to be
waited for. A value of X25anyevent may be used to accept
any event.

actevent is the event that actually occurred. It
consists of a ucepid and an event code.

acteventdata is additional data related to the event that occurred.

The contents are:

```

For connect indication/ confirmation:
    acteventdata(1) - number of bytes of user data
For data indication:
    acteventdata(1) - number of bytes of user data
    acteventdata(2) - flag settings
For More data:
    acteventdata(1) - number of packets confirmed
For expedited data indication:
    acteventdata(1) - value of first byte of data
For Reset indication:
    acteventdata(1) - Resetreason
For Disconnect indication:
    acteventdata(1) - Discreason
    acteventdata(2) - number of bytes of user data
For error indication:
    acteventdata(1) - Error return value
    acteventdata(2) - Primitive that caused the
                      error
For ports/queues not known by the library:
    acteventdata(1) - XMSG port number

```

If the events X25everin, X25evunkn, or X25evothr occur, X25PWAI will return immediately. These events override any requested event/CEP combination.

For X25everin events one of the following values is returned in acteventdata(2):

```

X25epatr : attach request
X25epcnr : connection request
X25epcna : connection response
X25epdcr : disconnect request
X25epdar : data request
X25epedr : expedited data request
X25epedi : expedited data indication
X25eprsr : reset request
X25eprsa : reset response

```

USAGE . . . : X25Pwai(timeout, regevent, retstat, actevent, acteventdata)

CHAPTER 3

SAMPLE PROGRAMS

3 SAMPLE PROGRAMS

```

=====
%
%                               X 2 5 - E X A M P L E - A
%
% This program initiates an X.25 connection to the program X25P-EXAMPLE-B,
% sends and receives some data, and then disconnects. For simplicity,
% error conditions are not handled properly.
%=====

MODULE X25_example_a

% Insert X.25 Definitions and Import files.
$LIST OFF
$INCLUDE X25P:DEFS
$INCLUDE X25P:IMPT
$LIST ON

% Define the stack.
INTEGER ARRAY: stack(0:1000)

PROGRAM: examplea

INTEGER ARRAY: mywork(0:500)      % Work area
X25Maxima: Maxes                  % Record containing max values for X25PINI
INTEGER: status                   % Return status
INTEGER: xid                      % Global id
INTEGER: pid                     % Port id
CONSTANT uid = 1111              % My id for the connection
INTEGER: cid                     % X25's id for the connection
X25Dtetype: remotenumber         % DTE number of the other side
X25outSVCconnection: outcondets  % outgoing connection conditions
X25DTEsuftype: suffix            % My DTE suffix
X25tm: waittime                  % Timeout when waiting
X25ev: regevent                  % Required event being waited for
X25ev: actevent                  % Actual event that occurred
INTEGER ARRAY: evdata(0:1)       % Data associated with actual event
X25nidusizes POINTER: confsizeptr % NIDU sizes returned by X25PATC
X25qresources POINTER: confqresptra % Queue resources returned by X25PATC
                                   % (always NIL)
X25facilities POINTER: facptr    % Facilities returned by X25PCNC
X25qs POINTER: Qosptr           % Quality of service returned by X25PCNC
                                   % (always NIL)
BYTES: udata(0:100)             % Buffer for user data
X25userdata: udataptr           % Record pointing to user data
INTEGER: udbufid                 % Buffer id returned by X25PDAI
                                   % (not used in this example)
INTEGER: flags                   % Flag bits returned by X25PDAI

INISTACK stack

%

```

```

% Initialize.
%
1 =: maxes.x25ports           % Only one X.25 module to be used
2 =: maxes.x25ceps           % Only one Connection End Point needed
X25Pini(mywork,maxes,status,xid)
IF status >< 0 THEN ENDIF % Handle error
%
% Open an association with X.25 module 0 on machine PLUTO.
%
X25Popn(xid,'PLUTO',0,nonqueued,X25defentity,status,pid)
IF status >< 0 THEN ENDIF % Handle error
%
% Set up a CEP for an Outgoing SVC.
%
X25SVCountgoing =: outcondets.X25conntype
ADDR(suffix) =: outcondets.X25DTEsuffix
X25DTEsub =: suffix.X25SAptype      % My suffix is DTE substring
2         =: suffix.X25suflength
'98'      =: suffix.X25DTEsubaddress
X25Patr(pid,uid,outcondets,TRUE,NIL,NIL,status,cid)
IF status >< 0 THEN ENDIF % Handle error
%
% Wait for an event for this CEP. Should be an attach confirmation.
%
X25infinitetime =: waittime.X25tmlength
X25anyevent =: regevent.X25evcode
cid         =: regevent.X25evcid
X25pwai(waittime,regevent,status,actevent,evdata)
IF actevent.X25evcode = X25evatcf THEN
    % Get info about the attach confirmation.
    X25patc(cid,status,confsizeptr,confqresp)
    IF status >< 0 THEN ENDIF % Handle error
ELSE
    % Some unexpected event - error
ENDIF
%
% Now request a connection via X.25 to the remote location.
% This should be running the program X25-EXAMPLE-B.
%
'234263500140 ' =: remotenumber.X25dtemainaddress
'99 '          =: remotenumber.X25dtesubaddress
0 =: udataptr.X25Udlength      % No user data
% Note that the addresses are terminated by a space.
X25pcnr(cid,remotenumber,NIL,NIL,udataptr,status)
IF status >< 0 THEN ENDIF % Handle error
%
% Wait for connect confirmation from the other side.
%
X25pwai(waittime,regevent,status,actevent,evdata)
IF actevent.X25evcode = X25evcncf THEN
    % Successful connection. Get the values returned by the other side.
    X25pcnc(cid,status,remotenumber,facptr,qosptr,udataptr)
    IF status >< 0 THEN ENDIF % Handle error
ELSE
    % Unexpected event - error.
ENDIF
%

```

```
% Connection is now set up. Send a message to the other side.
%
'Hello - how are you?' =: udata
ADDR(udata(0)) FORCE X25Intaddress =: udataptr.X25Udaddress
20 =: udataptr.X25Udlength
X25pdar(cid,0,udataptr,status)
IF status = 0 THEN
    OUTPUT(1,'A','$Message sent:')
    OUTPUT(1,'A',udata(0:udataptr.X25Udlength-1))
ELSE % Handle error
ENDIF
%
% Now wait for an event. This should be a message from the other side.
%
X25pwai(waittime,reqevent,status,actevent,evdata)
IF actevent.X25evcode = X25evdain THEN
    % Message received. Read it into a buffer.
    X25pdai(cid,status,flags,udataptr,udbufid)
    IF status = 0 THEN
        OUTPUT(1,'A','$Message received:')
        OUTPUT(1,'A',udata(0:udataptr.X25Udlength-1))
    ELSE % Handle error
    ENDIF
ELSE
    % Unexpected event - error.
ENDIF
%
% Disconnect the connection.
%
0 =: udataptr.X25Udlength % No user data
X25Pdcr(cid,0,udataptr,status)
IF status >< 0 THEN ENDIF % Handle error
%
% Tidy up by detaching the CEP, and closing the association
% with the X.25 module.
%
X25Pdtr(cid,status)
X25Pcls(pid,status)

ENDROUTINE

ENDMODULE
```

```
%=====
%
%               X 2 5 - E X A M P L E - B
%
% This program responds to an X.25 connection from the program
% X25P-EXAMPLE-A, receives some data, and then sends some back.
% For simplicity, error conditions are not handled properly.
%=====
```

```
MODULE X25_example_b
```

```
% Insert X.25 Definitions and Import files.
```

```
$LIST OFF
```

```
$INCLUDE X25P:DEFS
```

```
$INCLUDE X25P:IMPT
```

```
$LIST ON
```

```
% Define the stack.
```

```
INTEGER ARRAY: stack(0:1000)
```

```
PROGRAM: exampleb
```

```
INTEGER ARRAY: mywork(0:500)
```

```
X25Maxima: maxes
```

```
INTEGER: status
```

```
INTEGER: xid
```

```
INTEGER: pid
```

```
CONSTANT uid = 4321
```

```
INTEGER: cid
```

```
X25Dtetype: mynumber
```

```
X25Dtetype: remotenumber
```

```
X25inSVCconnection: incondets
```

```
X25DTEsuftype: suffix
```

```
X25tm: waittime
```

```
X25ev: regevent
```

```
X25ev: actevent
```

```
INTEGER ARRAY: evdata(0:1)
```

```
X25nidusizes POINTER: confsizeptr
```

```
X25qresources POINTER: confqresptr
```

```
X25facilities POINTER: Facptr
```

```
X25qs POINTER: Qosptr
```

```
BYTES: udata(0:31)
```

```
X25userdata: udataptr
```

```
INTEGER: udbufid
```

```
INTEGER: Flags
```

```
INISTACK stack
```

```
%
```

```
% Initialize
```

```
%
```

```
1 =: maxes.x25ports
```

```
1 =: maxes.x25ceps
```

```
X25Pini(mywork,maxes,status,xid)
```

```
% Work area
```

```
% Record containing max values for X25PINI
```

```
% Return status
```

```
% Global id
```

```
% Port id
```

```
% My id for the connection
```

```
% X25's id for the connection
```

```
% DTE number of this side
```

```
% DTE number of the other side
```

```
% Incoming Connection conditions
```

```
% My DTE suffix
```

```
% Timeout when waiting
```

```
% Required event being waited for
```

```
% Actual event that occurred
```

```
% Data associated with actual event
```

```
% NIDU sizes returned by X25PATC
```

```
% Queue resources returned by X25PATC
```

```
% (always NIL)
```

```
% Facilities returned by X25PCNC
```

```
% Quality of service returned by X25PCNC
```

```
% (always NIL)
```

```
% Buffer for user data
```

```
% Record pointing to user data
```

```
% Buffer id returned by X25PDAI (not used in this
```

```
% Flag bits returned by X25PDAI
```

```

IF status >< 0 THEN ENDIF %Handle error
%
% Open an association with X.25 module 0 on machine PLUTO.
%
X25Popn(xid,'PLUTO',0,nonqueued,X25defentity,status,pid)
IF status >< 0 THEN ENDIF %Handle error
%
% Set up a CEP for an Incoming SVC.
%
X25SVCincoming =: incondets.X25conntype
X25fourstep    =: incondets.X25connprotocol
ADDR(suffix) =: incondets.X25SAPsuffix
X25DTEsub =: suffix.X25SAPtype           % My suffix is DTE substring
2          =: suffix.X25suflength
'99'       =: suffix.X25DTEsubaddress
X25Patr(pid,uid,incondets,FALSE,NIL,NIL,status,cid)
IF status >< 0 THEN ENDIF %Handle error
%
% Wait for an event for this CEP. Should be an attach confirmation.
%
X25infinitetime =: waittime.X25tmlength
X25anyevent =: regevent.X25evcode
cid =: regevent.X25evid                % Wait for any CEP.
X25pwai(waittime,regevent,status,actevent,evdata)
IF actevent.X25evcode = X25evatcf THEN
    % Get info about the attach confirmation.
    X25patc(cid,status,confsizeptr,confqresp)
    IF status >< 0 THEN ENDIF %Handle error
ELSE
    % Some unexpected event - error
ENDIF
%
% Now wait for an incoming connection.
% This should come from the program X25-EXAMPLE-A.
%
X25pwai(waittime,regevent,status,actevent,evdata)
IF actevent.X25evcode = X25evcnin THEN
    % Incoming connection request.
    X25pcni(cid,status,mynumber,remotenum,Factr,Qosptr,udataptr)
    IF status >< 0 THEN ENDIF %Handle error
    % Send a connection response to confirm that it is OK.
    0 =: udataptr.X25Udlength           % No user data sent
    X25pcna(cid,mynumber,NIL,NIL,udataptr,status)
    IF status >< 0 THEN ENDIF %Handle error
ELSE
    %Handle error
ENDIF
%
% Connection is now set up. Now wait for an event.
% This should be a message from the other side.
%
X25pwai(waittime,regevent,status,actevent,evdata)
IF actevent.X25evcode = X25evdain THEN
    % Message received. Read it into a buffer.
    ADDR(udata(0)) FORCE X25Intaddress =: udataptr.X25Udaddress
    SIZE(udata) =: udataptr.X25udlength
    X25pdai(cid,status,Flags,udataptr,udbufid)

```

```
IF status = 0 THEN
    OUTPUT(1,'A','$Message received:')
    OUTPUT(1,'A',udata(0:udataptr.X25Udlength-1))
ELSE %Handle error
ENDIF
ELSE %Handle error
ENDIF
%
% Send a message to the other side.
%
'Very well thank you!' =: udata
20 =: udataptr.X25Udlength
X25pdar(cid,0,udataptr,status)
IF status = 0 THEN
    OUTPUT(1,'A','$Message sent:')
    OUTPUT(1,'A',udata(0:udataptr.X25Udlength-1))
ELSE %Handle error
ENDIF
%
% Now wait for disconnect indication.
% This should come from the program X25-EXAMPLE-A.
%
X25pwai(waittime,reqevent,status,actevent,evdata)
IF actevent.X25evcode >< X25evdcin THEN %Handle error
ENDIF
%
% Tidy up by detaching the CEP, and closing the association
% with the X.25 module.
%
X25Pdtr(cid,status)
X25Pcls(pid,status)

ENDROUTINE

ENDMODULE
```

A P P E N D I X A

ERROR CODES

General errors:

- > X25erilcs : Command illegal in this state
- > X25erbdid : Specified id is invalid
- > X25erplcg : Packet level congestion
- > X25erinac : Packet level inactive
- > X25erntcg : Network congestion
- > X25erpltm : Packet level timeout

Errors specific to commands:

- > X25erwksz : Workarea is too small (X25pini)
- > X25erptnm : Maximum number of ports exceeded (X25popn)
- > X25eralcd : Already listening for this condition (X25patr)
- > X25erforc : Event forced due to lack of event space (X25pwai).
The wait has been ended by an event that you were not waiting for,
because there is no space left in the work area for storing more
events.
- > X25ercpnm : Maximum number of CEPs exceeded (X25patr)
- > X25ercnm : Maximum number of connections exceeded (X25pcnr)
- > X25erbfm : Maximum number of Buffers exceeded (X25pprb)
- > X25erbfsz : Provided buffer too small (X25pdai). The buffer has
been filled with the incoming data, but some data has been lost.
- > X25ernocp : Message received for unknown CEP (X25pwai) -
internal error
- > X25erumsg : Unrecognized message from PL (X25pwai) - internal
error
- > X25ernocn : No connection available across the network
(X25pcnr)
- > X25erilpv : No PVC with this number (X25patr)
- > X25eratpv : PVC already attached (X25patr)
- > X25eralxd : Expedited data already sent but not yet confirmed
(X25pedr). X.25 does not allow more than one item of expedited data to
be sent at a time.

——→ X25ernoxd : No corresponding expedited indication (X25pedi). You have called the X25PEDI when there was no Expedited Data indication event.

——→ X25eralrs : Reset already sent but not yet confirmed (X25prsr). X.25 does not allow more than one reset to be done at a time.

——→ X25ernors : No corresponding reset indication (X25prsa). You have called X25PRSA when there was no reset indication event.

——→ X25erflag : Illegal data flags combination (X25pdar). When two or more packets are connected by use of the M-bit, they must have the same value of the Q-bit.

——→ X25erilud : Illegal user data length (X25pdar). The length specified is greater than the maximum length of a NIDU.

——→ X25ercrdt : Run out of credit for sending data (X25pdar). You must wait for an X25evcrdt event before sending more data.

Index

access point	3.
address	16, 20.
buffer	7, 38.
CEP	4, 19.
connection	
confirmation	5, 6, 24.
end point	4, 19.
indication	6, 25.
procedure 3-step	6.
procedure 4-step	6.
request	5, 6, 26.
response	6, 23.
disconnect	6, 8, 30.
event	9, 41, 42.
expedited data	8, 33.
network	
interface data unit	7, 27, 28.
service access point	3.
service data unit	7.
NIDU	7, 27, 28.
NSAP	3.
NSDU	7.
packet level	3.
permanent virtual circuit	4.
port	3.
PVC	4.
reset	8, 40.
SAP	3.
SVC	4.
switched virtual circuit	4.
virtual circuit	4.
x25patc	18.
X25patr	19.
X25pcls	22.
x25pcna	23.
x25pcnc	24.
x25pcni	25.
x25pcnr	26.
x25pdai	27.
X25pdar	28.
x25pdci	29.
x25pdcr	30.
x25pdtr	31.
x25pedi	32.
x25pedr	33.
x25pini	34.
X25popn	36.
x25pprb	38.
x25prsa	39.
x25prsr	40.
x25pwai	41.

SEND US YOUR COMMENTS!!!



Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card — and an answer to your comments.

Please let us know if you

- find errors
- cannot understand information
- cannot find information
- find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!



HELP YOURSELF BY HELPING US!!

Manual name: COSMOS X.25 Programmer Guide

Manual number: ND-60.227.1 EN

What problems do you have? (use extra pages if needed)

Do you have suggestions for improving this manual ?

Your name:

Date:

Company:

Position:

Address:

What are you using this manual for ?

NOTE!

This form is primarily for documentation errors. Software and system errors should be reported on Customer System Reports.

Send to:

Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Norsk Data's answer will be found on reverse side



Answer from Norsk Data

Answered by

Date

Norsk Data A.S

Documentation Department
P.O. Box 25, Bogerud
0621 Oslo6, Norway

