1.1 Enhancements

Message Set Support
Layer 3 message sets are now distributed as independent files that can be loaded by the ISDN Monitor or Emulation applications. Message sets are now selected on the Message Set Selection Menu under the MessageSet topic.

Report Format
In complete display format, octets are numbered by octet identifier (eg. octet 3a) instead of an incrementing number. String parameters can be displayed in text format using ASCII, EBCDIC, or hex character sets. The call reference value is padded with leading 00 bytes (if necessary) to indicate the call reference length.

Message Builder
The message builder now supports any defined layer 3 message set. The IE Selection Menu displays mandatory, optional, and other IE's for each message on separate screens. Undefined values can be specified on one screen to indicate which entries use two pool buffers. The Insert function key has been added to the buffer editor.

Test Script Message Builder
Temporary IE buffers can now be used in automatic mode when building messages from within test scripts. Information elements, from more than two different codesets, can be built for a single message.

X.25 Support
SAPI 16 I frame information is decoded according to the X.25 (1984) Recommendation. Filter and trigger support for X.25 packets has also been added.

State Machine Behaviour with Continuous Mode Enabled
When continuous queuing procedure is selected and the emulation state machine enters state 8 (timer recovery state) and returns to state 7 (multi frame operation), data transmission now resumes automatically.

National Message Set Support
Using MDL (message description language), all message sets, including the latest national message sets, are now supported in monitor and layer 3 simulation.

Basic Rate Emulation
X.25 PLP (packet layer procedure) emulation is now included (i.e. both keyboard and test script operation).
1.2 Changes

- **Monitor Configuration Menu**
  Frame Sequence Number Modulo has been replaced by Modulus Mode. Valid selections are NORMAL and EXTENDED instead of MOD 8 and MOD 128.

- **Test Script Incompatibility**
  Due to changes in the message builder, some test scripts that start by selecting a specific message set from within a test manager state, are not compatible with this release.

  Workaround:
  The LOAD_MESSAGE_SET command must be included prior to defining any of the test manager states.

  Example:
  In the AT&T test script ATT_NET.F, the AT&T message set is loaded by using the LOAD_MESSAGE_SET command.

  TCLR
  " ATT_5E6" LOAD_MESSAGE_SET
  " ATT.P" LOAD_MESSAGES

1.3 Problems Fixed

- **Trace Format**
  When trace display format is selected, data is no longer displayed along with trace statements.

- **Send Data Source Menu**
  The transmit mode on the Send Data Source Menu is now correctly displayed as continuous when the CONT_ON command is executed.

- **Message Builder Values**
  The MessageBuilder topic is no longer related to connection data structures (eg. *COD). Thus, changing the protocol discriminator, call reference, selected information elements, and parameter values has no effect on simulation scripts that use connections.

- **Data Source Menu**
  Messages selected from the Send Data Source Menu and transmitted from the Send topic are now correct for the currently selected message set.

- **Selective Playback**
  The data file header of a multi-channel data recording is now displayed during playback, even if a single channel of the recording is selected for display.

- **Bit Rate (WAN)**
  Transmitting continuous information frames at speeds less than 1800 bps no longer results in merged frames.
External Clock (WAN)

When the tester is configured as User and clocking is provided by the interface (External Clock Off), the bit rate can only be measured. When clocking is provided by the tester (External Clock On), the bit rate must be selected.

When the tester is configured as Network and clocking is provided by the interface (External Clock On), the bit rate can only be measured. When clocking is provided by the tester (External Clock Off), the bit rate must be selected.

S/T Bus Power Supply (Basic Rate)

The PS1 and PS2 commands, which controlled the application of power supplies, are no longer interchanged (i.e. PS1 commands controlling PS2 and vice versa). Consequently, test scripts written to control these power supplies must be similarly modified.

State Traces (Basic Rate & WAN)

Layer 2 states 1, 2, 3, and 4 are now reported during the TEI assignment procedures after the STATE_ON command is executed.

TEI User Request (Basic Rate & WAN)

When two (or more) user applications request TEI's from the network simultaneously, two different reference numbers (Ri) in the ID request message are randomly assigned.

Layer 1 Filters and Triggers (Primary Rate)

Layer 1 filters and triggers for facility error indications have been removed from the Layer 1 Filter Setup Menu and the Layer 1 Event Menu.

Monitoring High Throughput Timeslots (Primary Rate)

The tester no longer hangs-up or produces a bus error when monitoring high throughput data channels in complete display format.

1.4 Known Problems

C/R Bit Settings

The layer 2 emulation can incorrectly accept frames that have a C/R bit error.

Triggering

If a trigger is set up for both a data window display message and highlighting, only the data window display message will be displayed; no highlighting occurs.

Selective Playback

If data is captured to disk midway through another capture being performed by a different channel, data may appear between the two lines composing the header, indicating the start of the second capture to disk.

Interface Speed (WAN)

The interface speed 128000 b/s is not supported.
V.36 (RS-449) Interface (WAN)
In Emulation mode, when an unterminated cable is connected to the V.36 interface, changing the emulation type (eg. TO DTE or TO DCE) can cause the tester to lock-up.

Workaround:
Disconnect the cable, select the emulation type, and then reconnect the cable.

NOTE
Connecting cable(s) of any other type (i.e. V.35, V.11 etc.) or selecting any interface type from the Configuration Menu will not cause a lock up if a V.36 physical cable is not connected.

When the emulation is running in the background, an incorrect result is displayed when trying to measure response times in playback RAM.

The maximum length of the pool entry name on the Message Pool Menus is longer than the Overview Window can display, resulting in truncated names on the Overview Menu.

Workaround:
The Next or Previous function keys must be used to scroll through all messages. This displays the current name used in the Pool Entry item.

When displaying messages within the message builder and complete report format is selected, lines containing RED FG graphic characters are displayed after scrolling down and back up again. This only occurs when the emulation is running in background.
This manual is intended to provide a programmer's guide to the ISDN Monitor/Emulation programs. General programming information is provided in the Programmer's Reference Manual. Information contained in this manual is machine independent.

This manual is not intended to provide basic user instruction, but rather addresses the issues of writing test programs using the Interactive Test Language (ITL). Refer to the machine specific User Manual for a quick reference to the basic operation of the protocol tester.

IDACOM reserves the right to make any required changes in this manual without prior notice, and the user should contact IDACOM to determine if any changes have been made. No part of this manual may be photocopied, reproduced, or translated without the prior written consent of IDACOM.

IDACOM makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.
# TABLE OF CONTENTS

**PREFACE**

1. **INTRODUCTION** .......................................................... 1-1

2. **MONITOR ARCHITECTURE** ............................................. 2-1
   2.1 Live Data .............................................................. 2-1
   2.2 Playback ............................................................. 2-2
      Playback Control .................................................. 2-3
   2.3 Simultaneous Live Data and Playback .......................... 2-4

3. **MONITOR CONFIGURATION** ............................................ 3-1
   3.1 Layer 1 ............................................................... 3-1
      Basic Rate ......................................................... 3-1
      Primary Rate ..................................................... 3-3
      WAN ................................................................. 3-4
   3.2 Layer 2 ............................................................... 3-5

4. **CAPTURE RAM** .......................................................... 4-1
   4.1 Capturing to RAM .................................................. 4-1
   4.2 Transferring from RAM .......................................... 4-2
      To Disk .............................................................. 4-3
      To Printer .......................................................... 4-4

5. **DISK RECORDING** ..................................................... 5-1

6. **DISPLAY FORMAT** .................................................... 6-1

7. **FILTERS** ............................................................... 7-1
   7.1 Layer 1 ............................................................... 7-1
   7.2 Layer 2 ............................................................... 7-5
   7.3 Layer 3 ............................................................... 7-7
   7.4 X.25 Layer 3 ........................................................ 7-9

---

IDACOM

ISDN Programmer's Manual
# TABLE OF CONTENTS [continued]

8 TRIGGERS ........................................................................................................... 8-1
  8.1 Trigger Control ......................................................................................... 8-1
  8.2 Trigger Conditions .................................................................................. 8-2
      Layer 1 ........................................................................................................... 8-3
      Layer 2 ........................................................................................................... 8-4
      Layer 3 ........................................................................................................... 8-6
      X.25 Layer 3 ................................................................................................... 8-7
      Other Triggers ............................................................................................. 8-8
  8.3 Trigger Actions ............................................................................................ 8-9

9 LAYER 1 DECODER ............................................................................................ 9-1

10 LAYER 2 DECODER .......................................................................................... 10-1
   10.1 FRMR Frames ......................................................................................... 10-3
   10.2 UI Frames ................................................................................................ 10-4
   10.3 XID Frames ............................................................................................. 10-4

11 LAYER 3 DECODER .......................................................................................... 11-1
   11.1 Variables ................................................................................................ 11-1
   11.2 Commands ............................................................................................... 11-3

12 X.25 LAYER 3 DECODER .................................................................................. 12-1

13 MESSAGE SETS .................................................................................................. 13-1

14 CCITT_1988 ..................................................................................................... 14-1
   14.1 Using Message Identifiers ........................................................................ 14-1
   14.2 Using IE Identifiers .................................................................................. 14-2
   14.3 Using IE Structures ................................................................................... 14-3
   14.4 Message Identifiers ................................................................................... 14-4
       Q.931 Protocol Discriminator (value = 0X08) ........................................... 14-4
   14.5 IE Identifiers .............................................................................................. 14-5
       Codeset 0 ..................................................................................................... 14-5
       Codeset 5 ..................................................................................................... 14-6
# TABLE OF CONTENTS [continued]

## 14 CCITT_1988 [continued]

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Codeset 6</td>
<td>14-6</td>
</tr>
<tr>
<td>Codeset 7</td>
<td>14-6</td>
</tr>
</tbody>
</table>

### 14.6 IE Structures

- Bearer Capability IE (l#BEARER_CAP) | 14-6
- Called Party Number IE (l#CALLED_NUM) | 14-6
- Called Party Subaddress IE (l#CALLED_SAD) | 14-10
- Calling Party Number IE (l#CALLING_NUM) | 14-11
- Calling Party Subaddress IE (l#CALLING_SAD) | 14-11
- Call Identity IE (l#CALL_ID) | 14-12
- Call State IE (l#CALL_STATE) | 14-12
- Cause IE (l#CAUSE) | 14-13
- Channel Identification IE (l#CHANNEL_ID) | 14-14
- Congestion Level IE (l#CONG_LEVEL) | 14-15
- Date/time IE (l#DATE/TIME) | 14-15
- Display IE (l#DISPLAY) | 14-16
- End-to-end Transit Delay IE (l#END-END_DELAY) | 14-16
- Endpoint Identifier IE (l#ENDPOINT_ID) | 14-16
- Facility IE (l#FACILITY) | 14-17
- Feature Activation IE (l#FEAT_ACT) | 14-17
- Feature Indication IE (l#FEAT_IND) | 14-18
- High Layer Compatibility IE (l#HL_LAY_COMP) | 14-18
- Information Rate IE (l#INFO_RATE) | 14-19
- Information Request IE (l#INFO_REQ) | 14-20
- Keypad IE (l#KEYPAD) | 14-20
- Low Layer Compatibility IE (l#LOW_LAY_COMP) | 14-21
- More Data IE (l#MORE_DATA) | 14-24
- Network Facilities IE (l#NETWORK_FACIL) | 14-25
- Notification Indicator IE (l#NOTIFIC_IND) | 14-25
- Packet Layer Binary Parameters IE (l#PKT_PARAMS) | 14-25
- Packet Size IE (l#PKT_SIZE) | 14-26
- Packet Layer Window Size IE (l#PKT_WINDOW) | 14-26
- Progress Indicator IE (l#PROGRESS_IND) | 14-26
- Redirecting Number IE (l#REDIRING_NUM) | 14-27
- Repeat Indicator IE (l#REPEAT_IND) | 14-28
- Restart Indicator IE (l#RESTART_IND) | 14-28
- Segmented Message IE (l#SEGMENT) | 14-28
- Sending Complete IE (l#SEND_COMP) | 14-29
- Shift IE (l#SHIFT) | 14-30
- Signal IE (l#SIGNAL) | 14-30
- Service Profile Id. IE (l#SPID) | 14-30
- Switchhook IE (l#SWITCHHOOK) | 14-31
- Transit Delay Selection IE (l#TRANS_DEL_SEL) | 14-31
- Transit Network Select IE (l#TRANS_NW_SEL) | 14-31
- User-user Information IE (l#UU_INFO) | 14-32

## 15 EMULATION ARCHITECTURE

### 15.1 Live Data

- Page 15-2

### 15.2 Playback

- Page 15-3

### 15.3 Simultaneous Live Data and Playback

- Page 15-4
### TABLE OF CONTENTS [continued]

#### 16 EMULATION CONFIGURATION

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.1 Layer 1</td>
<td>16–2</td>
</tr>
<tr>
<td></td>
<td>Basic Rate</td>
</tr>
<tr>
<td></td>
<td>Primary Rate</td>
</tr>
<tr>
<td></td>
<td>WAN</td>
</tr>
<tr>
<td>16.2 Layer 2</td>
<td>16–12</td>
</tr>
<tr>
<td></td>
<td>Link Setup</td>
</tr>
<tr>
<td></td>
<td>Management Layer/Assignment Source Procedure Setup</td>
</tr>
</tbody>
</table>

#### 17 LAYER 2 EMULATION

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.1 State-Independent Commands</td>
<td>17–1</td>
</tr>
<tr>
<td></td>
<td>Command/Response Bits</td>
</tr>
<tr>
<td></td>
<td>State Variables</td>
</tr>
<tr>
<td></td>
<td>FRMR Bits</td>
</tr>
<tr>
<td></td>
<td>Retransmission Counters</td>
</tr>
<tr>
<td></td>
<td>Miscellaneous</td>
</tr>
<tr>
<td>17.2 In-State Commands</td>
<td>17–7</td>
</tr>
<tr>
<td></td>
<td>State Machine Control</td>
</tr>
<tr>
<td></td>
<td>Services</td>
</tr>
<tr>
<td></td>
<td>Send Commands</td>
</tr>
<tr>
<td>17.3 Out-of-State Commands</td>
<td>17–14</td>
</tr>
<tr>
<td></td>
<td>Send Commands</td>
</tr>
<tr>
<td>17.4 Transmit Mode</td>
<td>17–17</td>
</tr>
<tr>
<td>17.5 Frame Errors</td>
<td>17–18</td>
</tr>
</tbody>
</table>

#### 18 LAYER 3 SIMULATION

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.1 Layer 3 Data Source</td>
<td>18–1</td>
</tr>
<tr>
<td>18.2 Connection Management</td>
<td>18–3</td>
</tr>
<tr>
<td></td>
<td>Connection Related Data Management</td>
</tr>
<tr>
<td></td>
<td>Call Reference Management</td>
</tr>
<tr>
<td></td>
<td>Information Element Management</td>
</tr>
<tr>
<td></td>
<td>Connection Timer Management</td>
</tr>
<tr>
<td>18.3 B-Channel Management</td>
<td>18–10</td>
</tr>
<tr>
<td>18.4 Message Builder</td>
<td>18–11</td>
</tr>
<tr>
<td></td>
<td>Generating Messages</td>
</tr>
<tr>
<td></td>
<td>Generating Information Elements</td>
</tr>
<tr>
<td>18.5 Message Buffer Pool Management</td>
<td>18–20</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

19 X.25 LAYER 3 PLP EMULATION ............................................. 19–1
  19.1 Emulation Configuration .................................................. 19–1
      Packet Layer ............................................................... 19–2
      Facilities ................................................................. 19–7
      LCN Setup ................................................................. 19–13
  19.2 Emulation Decode .......................................................... 19–19
      Communication Variables ................................................. 19–19
  19.3 Emulation Response ....................................................... 19–26
      Emulation State Machines .............................................. 19–27
      Automatic Responses ...................................................... 19–28
      State–Dependent Send Commands ....................................... 19–29
      State–Independent Send Commands ...................................... 19–32

20 TEST MANAGER ................................................................. 20–1
  20.1 ITL Constructs .............................................................. 20–1
  20.2 Control Commands .......................................................... 20–3
  20.3 Event Recognition .......................................................... 20–3
      Layer 1 ...................................................................... 20–4
      Layer 2 ...................................................................... 20–4
      Layer 3 ...................................................................... 20–6
      X.25 Layer 3 .............................................................. 20–7
      Timeout Detection ........................................................ 20–10
      Function Key Detection .................................................. 20–12
      Interprocessor Mail Events .............................................. 20–12
      Wildcard Events .......................................................... 20–12
  20.4 ISDN Actions ............................................................... 20–13
      Layer 1 Actions ........................................................... 20–13
      Lapse Timers .............................................................. 20–14
      Interval Timers ........................................................... 20–15
      Using Buffers ............................................................. 20–15
  20.5 X.25 PLP Actions ............................................................ 20–17
  20.6 Layer 3 Connection Multiplexer ....................................... 20–18

21 TEST SCRIPTS ................................................................. 21–1
  21.1 DMS_1.F ................................................................. 21–2
  21.2 DMS_2 ................................................................. 21–8
      DMS_2a.F ............................................................... 21–8
      DMS_2b.F ............................................................... 21–13
  21.3 ATT_NET.F .............................................................. 21–21
      Message Pool Descriptions .............................................. 21–24
### TABLE OF CONTENTS [continued]

#### 21 TEST SCRIPTS [continued]

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.4</td>
<td>1TR6 Primary Rate Test Scripts</td>
<td>21-29</td>
</tr>
<tr>
<td></td>
<td>S2M_N.F</td>
<td>21-29</td>
</tr>
<tr>
<td></td>
<td>S2M_U.F</td>
<td>21-37</td>
</tr>
<tr>
<td></td>
<td>Message Pool Descriptions</td>
<td>21-42</td>
</tr>
<tr>
<td>21.5</td>
<td>Layer 3 Simulation</td>
<td>21-47</td>
</tr>
<tr>
<td></td>
<td>L3SIM.F</td>
<td>21-47</td>
</tr>
<tr>
<td></td>
<td>L3S_NTWK.F</td>
<td>21-52</td>
</tr>
<tr>
<td></td>
<td>L3S_USER.F</td>
<td>21-60</td>
</tr>
<tr>
<td>21.6</td>
<td>X.25 PLP Monitor/Emulation</td>
<td>21-68</td>
</tr>
<tr>
<td></td>
<td>TEST1_X25.F</td>
<td>21-68</td>
</tr>
<tr>
<td></td>
<td>TEST2_X25.F</td>
<td>21-71</td>
</tr>
<tr>
<td></td>
<td>X25P_STAT.F</td>
<td>21-75</td>
</tr>
</tbody>
</table>

#### APPENDICES

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>INTRODUCTION TO ISDN</td>
<td>A-1</td>
</tr>
<tr>
<td>A.1</td>
<td>The Origins of ISDN</td>
<td>A-1</td>
</tr>
<tr>
<td>A.2</td>
<td>ISDN Access</td>
<td>A-1</td>
</tr>
<tr>
<td>A.3</td>
<td>The ISDN Network</td>
<td>A-3</td>
</tr>
<tr>
<td>A.4</td>
<td>Channel Structure</td>
<td>A-5</td>
</tr>
<tr>
<td>A.5</td>
<td>Communication Between Layers in the D-Channel</td>
<td>A-6</td>
</tr>
<tr>
<td>A.6</td>
<td>Layer 2 – Data Link Layer (LAPD)</td>
<td>A-7</td>
</tr>
<tr>
<td></td>
<td>Addressing</td>
<td>A-7</td>
</tr>
<tr>
<td></td>
<td>Establishing a Layer 2 Link</td>
<td>A-8</td>
</tr>
<tr>
<td></td>
<td>LAPD Frames</td>
<td>A-10</td>
</tr>
<tr>
<td></td>
<td>LAPD Frame Structure</td>
<td>A-12</td>
</tr>
<tr>
<td>A.7</td>
<td>Layer 3 – Network Layer</td>
<td>A-12</td>
</tr>
<tr>
<td></td>
<td>Operation of the Network Layer</td>
<td>A-13</td>
</tr>
<tr>
<td></td>
<td>Structure of the Network Layer</td>
<td>A-14</td>
</tr>
<tr>
<td>B</td>
<td>INTRODUCTION TO CCITT X.25 LAYER 3</td>
<td>B-1</td>
</tr>
<tr>
<td>B.1</td>
<td>Purpose of the Recommendation</td>
<td>B-1</td>
</tr>
<tr>
<td>B.2</td>
<td>Types of X.25 Services</td>
<td>B-1</td>
</tr>
<tr>
<td>B.3</td>
<td>Layer 3 – Network Level</td>
<td>B-1</td>
</tr>
<tr>
<td></td>
<td>GFI (General Format Identifier)</td>
<td>B-2</td>
</tr>
<tr>
<td></td>
<td>Logical Channel Group Number and Logical Channel Number</td>
<td>B-2</td>
</tr>
<tr>
<td></td>
<td>Packet Identifier</td>
<td>B-3</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS [continued]

**B** INTRODUCTION TO CCITT X.25 LAYER 3 [continued]

- B.4 Example of Call Setup, Data Transfer, and Call Clear  
  \[ B-4 \]

**C** XID PROCEDURES  
\[ C-1 \]

**D** CODING CONVENTIONS  
\[ D-1 \]

- D.1 Stack Comments  
  \[ D-1 \]
- D.2 Stack Comment Abbreviations  
  \[ D-2 \]
- D.3 Program Comments  
  \[ D-2 \]
- D.4 Test Manager Constructs  
  \[ D-2 \]
- D.5 Spacing and Indentation Guidelines  
  \[ D-3 \]
- D.6 Colon Definitions  
  \[ D-4 \]

**E** ASCII/EBCDIC/HEX CONVERSION TABLE  
\[ E-1 \]

**F** COMMAND CROSS REFERENCE LIST  
\[ F-1 \]

INDEX
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Sample Stack Comment</td>
<td>1-2</td>
</tr>
<tr>
<td>2-1</td>
<td>ISDN Monitor Data Flow Diagram – Live Data</td>
<td>2-1</td>
</tr>
<tr>
<td>2-2</td>
<td>ISDN Monitor Data Flow Diagram – Offline Processing</td>
<td>2-2</td>
</tr>
<tr>
<td>2-3</td>
<td>ISDN Monitor Data Flow Diagram – Freeze Mode</td>
<td>2-4</td>
</tr>
<tr>
<td>3-1</td>
<td>Monitor Configuration Menu (WAN)</td>
<td>3-4</td>
</tr>
<tr>
<td>3-2</td>
<td>Monitor Configuration Menu (Basic Rate)</td>
<td>3-5</td>
</tr>
<tr>
<td>4-1</td>
<td>ISDN Data Flow Diagram – Capture to RAM</td>
<td>4-1</td>
</tr>
<tr>
<td>5-1</td>
<td>ISDN Data Flow Diagram – Recording to Disk</td>
<td>5-1</td>
</tr>
<tr>
<td>6-1</td>
<td>ISDN Data Flow Diagram – Display and Print</td>
<td>6-1</td>
</tr>
<tr>
<td>6-2</td>
<td>Display Format Menu</td>
<td>6-2</td>
</tr>
<tr>
<td>7-1</td>
<td>Layer 1 Filter Setup Menu (Basic Rate)</td>
<td>7-1</td>
</tr>
<tr>
<td>7-2</td>
<td>Layer 1 Filter Setup Menu (Primary Rate)</td>
<td>7-3</td>
</tr>
<tr>
<td>7-3</td>
<td>Layer 2 Filter Setup Menu</td>
<td>7-5</td>
</tr>
<tr>
<td>7-4</td>
<td>Layer 3 Filter Setup Menu</td>
<td>7-7</td>
</tr>
<tr>
<td>8-1</td>
<td>Layer 1 Event Menu (Basic Rate)</td>
<td>8-3</td>
</tr>
<tr>
<td>8-2</td>
<td>Layer 2 Event Menu</td>
<td>8-4</td>
</tr>
<tr>
<td>8-3</td>
<td>Layer 3 Event Menu</td>
<td>8-6</td>
</tr>
<tr>
<td>8-4</td>
<td>Packet Reception Triggers</td>
<td>8-7</td>
</tr>
<tr>
<td>8-5</td>
<td>Trigger Action Menu</td>
<td>8-9</td>
</tr>
<tr>
<td>15-1</td>
<td>ISDN Emulation Data Flow Diagram – Live Data</td>
<td>15-2</td>
</tr>
<tr>
<td>15-2</td>
<td>ISDN Emulation Data Flow Diagram – Offline Processing</td>
<td>15-3</td>
</tr>
<tr>
<td>15-3</td>
<td>ISDN Emulation Data Flow Diagram – Freeze Mode</td>
<td>15-4</td>
</tr>
<tr>
<td>16-1</td>
<td>Selfloop Configuration Diagram</td>
<td>16-6</td>
</tr>
<tr>
<td>16-2</td>
<td>Crossloop Configuration Diagram</td>
<td>16-7</td>
</tr>
<tr>
<td>16-3</td>
<td>B1 to External 1, B2 to Voice Configuration Diagram</td>
<td>16-8</td>
</tr>
<tr>
<td>16-4</td>
<td>Emulation Configuration Menu (WAN)</td>
<td>16-9</td>
</tr>
<tr>
<td>16-5</td>
<td>Emulation Configuration Menu (Basic Rate)</td>
<td>16-12</td>
</tr>
<tr>
<td>16-6</td>
<td>Link Setup Menu</td>
<td>16-15</td>
</tr>
<tr>
<td>16-7</td>
<td>Management Layer Setup Menu</td>
<td>16-20</td>
</tr>
<tr>
<td>16-8</td>
<td>Assignment Source Procedure Menu</td>
<td>16-22</td>
</tr>
<tr>
<td>17-1</td>
<td>Alert Message</td>
<td>17-12</td>
</tr>
<tr>
<td>17-2</td>
<td>Call Proceeding Message</td>
<td>17-13</td>
</tr>
<tr>
<td>19-1</td>
<td>Setup Menu</td>
<td>19-1</td>
</tr>
<tr>
<td>19-2</td>
<td>DTE Packet Layer Menu</td>
<td>19-3</td>
</tr>
<tr>
<td>19-3</td>
<td>DCE Packet Layer Menu</td>
<td>19-5</td>
</tr>
<tr>
<td>19-4</td>
<td>Facility Menu</td>
<td>19-7</td>
</tr>
<tr>
<td>19-5</td>
<td>LCN Setup Menu 1</td>
<td>19-13</td>
</tr>
<tr>
<td>19-6</td>
<td>LCN Setup Menu 2</td>
<td>19-16</td>
</tr>
<tr>
<td>20-1</td>
<td>Buffer Structure</td>
<td>20-15</td>
</tr>
<tr>
<td>A-1</td>
<td>Basic Rate Configuration</td>
<td>A-2</td>
</tr>
<tr>
<td>A-2</td>
<td>Primary Rate Configuration</td>
<td>A-2</td>
</tr>
<tr>
<td>A-3</td>
<td>ISDN Network</td>
<td>A-4</td>
</tr>
<tr>
<td>A-4</td>
<td>OSI Model of the D-Channel</td>
<td>A-5</td>
</tr>
<tr>
<td>A-6</td>
<td>TEI/SAPI Relationship</td>
<td>A-8</td>
</tr>
<tr>
<td>A-7</td>
<td>Data Flow in a Link Establishment</td>
<td>A-9</td>
</tr>
<tr>
<td>A-8</td>
<td>LAPD Frame Structure</td>
<td>A-12</td>
</tr>
<tr>
<td>A-9</td>
<td>ISDN Voice Communication</td>
<td>A-13</td>
</tr>
<tr>
<td>A-10</td>
<td>Layer 2 Information Frames</td>
<td>A-14</td>
</tr>
<tr>
<td>A-11</td>
<td>Network Layer Packet</td>
<td>A-14</td>
</tr>
<tr>
<td>A-12</td>
<td>Single Octet Information Element</td>
<td>A-15</td>
</tr>
<tr>
<td>A-13</td>
<td>Variable Length Information Element</td>
<td>A-15</td>
</tr>
<tr>
<td>B-1</td>
<td>Call Setup, Data Transfer, and Call Clearing</td>
<td>B-4</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6-1</td>
<td>Dual Window Commands</td>
</tr>
<tr>
<td>7-1</td>
<td>Basic Rate Layer 1 Filter Commands</td>
</tr>
<tr>
<td>7-2</td>
<td>Primary Rate Layer 1 Filter Commands</td>
</tr>
<tr>
<td>7-3</td>
<td>Layer 2 Filter Commands</td>
</tr>
<tr>
<td>7-4</td>
<td>Selective Filter Commands</td>
</tr>
<tr>
<td>7-5</td>
<td>X.25 Layer 3 Filter Commands</td>
</tr>
<tr>
<td>8-1</td>
<td>Basic Rate Layer 1 Trigger Commands</td>
</tr>
<tr>
<td>8-2</td>
<td>Primary Rate Layer 1 Trigger Commands</td>
</tr>
<tr>
<td>8-3</td>
<td>Layer 2 Trigger Commands</td>
</tr>
<tr>
<td>8-4</td>
<td>X.25 Layer 3 Trigger Commands</td>
</tr>
<tr>
<td>13-1</td>
<td>Message Set Filename Extensions – Monitor Applications</td>
</tr>
<tr>
<td>13-2</td>
<td>Message Set Filename Extensions – Emulation Applications</td>
</tr>
<tr>
<td>16-1</td>
<td>Effect of Clocking and Emulation Interface Selections on Bit Rate</td>
</tr>
<tr>
<td>17-1</td>
<td>Layer 2 States</td>
</tr>
<tr>
<td>17-2</td>
<td>Management States</td>
</tr>
<tr>
<td>17-3</td>
<td>ASP States</td>
</tr>
<tr>
<td>18-1</td>
<td>Send Commands</td>
</tr>
<tr>
<td>19-1</td>
<td>Protocol Standard Parameters/Procedures</td>
</tr>
<tr>
<td>19-2</td>
<td>Layer 3 States</td>
</tr>
<tr>
<td>19-3</td>
<td>Logical Channel States</td>
</tr>
<tr>
<td>20-1</td>
<td>Packet Identifiers</td>
</tr>
<tr>
<td>20-2</td>
<td>V.28/RS–232C Interface Lead Transitions</td>
</tr>
<tr>
<td>20-3</td>
<td>V.35 Interface Lead Transitions</td>
</tr>
<tr>
<td>20-4</td>
<td>V.36/RS–449 Interface Lead Transitions</td>
</tr>
<tr>
<td>A-1</td>
<td>Frames for Modulo 8 Operation</td>
</tr>
<tr>
<td>A-2</td>
<td>Frames for Modulo 128 Operation</td>
</tr>
<tr>
<td>B-1</td>
<td>Packet Header</td>
</tr>
<tr>
<td>B-2</td>
<td>Packet Type Identifier</td>
</tr>
<tr>
<td>D-1</td>
<td>ITL Symbols</td>
</tr>
</tbody>
</table>
The ISDN Monitor is implemented in accordance with the following specifications:

Layer 2:
- Link Establish, Link Release and Data Transfer Phases
- TEI Assignment Procedure
  CCITT Rec. Q.921 COM XI Contribution 40–E, July 1985
  (Boulder, Colorado)
- XID Procedure
- DL–ESTABLISH–CONFIRM and DL–RELEASE–CONFIRM primitives
  CCITT Rec. Q.921 COM XI Contribution 40–E, July 1985
  (Boulder, Colorado)

Layer 3:
- CCITT Message Decoding
  CCITT Rec. Q.931 Blue Book, Nov. 1988
The ISDN Monitor is not a state-driven monitor, i.e. it does not have the knowledge of expected events. Rather, it decodes the data and reports information on received layer 1 events, frames, and messages. Filters, triggers, RAM capture, and disk recording are also available.

The ISDN layer 2 emulation is a state-driven protocol emulation together with an integral protocol monitor. The emulation can be driven as:

- an automatic emulation;
- a semi-automatic tester. The test manager is used to build and execute test scenarios.
- a manual tester. The test is controlled from the user's keyboard.

The layer 2 emulation can emulate on up to eight links simultaneously.

The layer 3 simulation provides the tools necessary to build and transmit or receive and decode layer 3 messages. These tools are:

- menu driven message building utility for building, editing and saving layer 3 messages;
- menu driven selection of default layer 3 messages; and
- test manager tools for constructing layer 3 messages inside a test script.

The layer 3 simulation can maintain the data associated with eight separate calls simultaneously.

All user test scripts are written in the ITL language. Test programs are made up of sequences of ITL commands (or 'words') which exchange data and parameters via a Last In First Out (LIFO) stack. All commands consume zero or more parameters from the stack (input) and/or leave results on the stack (output). These commands have a stack effect comment shown beside the definition of the command to define its input and output parameters.

![Sample Stack Comment](image)

\[ \text{Input Parameters} \quad \text{Output Parameters} \]
\[ ( \text{Par1} \ \text{Par2} \quad \text{Par3} \ \text{Par4} \ \text{Par5} ) \]
\[ \text{Item on top of stack} \quad \text{Item on top of stack} \]
\[ \text{Input/Output Separator} \]

\[ \text{Figure 1-1 Sample Stack Comment} \]

\[ \text{NOTE} \]

*See Appendix B for further explanation of stack parameters.*
The ISDN D-Channel Monitor program monitors live data, saves data to capture RAM or disk, and displays data in a number of different formats. Data can be passed through filters which limit the displayed, captured, or recorded data. Triggers perform specific actions when a specified event occurs.

### 2.1 Live Data

The monitor application receives events from the interface or from the internal timer and processes them as shown in Figure 2-1.

![Figure 2-1 ISDN Monitor Data Flow Diagram – Live Data](image)

**Display** topic

*Live Data* function key

**MONITOR** ( -- )

Selects the live data mode of operation. All incoming events are decoded and displayed in real-time.
2.2 Playback

Data (both protocol and lead information) can be examined in an offline mode using either capture RAM or a disk file as the data source.

Figure 2-2 ISDN Monitor Data Flow Diagram – Offline Processing

- **FROM_CAPT HALT**
  - Display topic
  - Playback RAM function key

- **FROM_DISK HALT PLAYBACK**
  - Display topic
  - Playback Disk function key

**HALT (--)**

Selects the playback mode of operation. Data is retrieved from capture RAM or a disk file, decoded, and displayed or printed. Capture RAM is suspended in this mode.

**FROM_CAPT (--)**

Selects the capture buffer as the source for data transfer.

**FROM_DISK (--)**

Selects a disk file as the source for data transfer.
PLAYBACK ( -- )

Opens a data recording file for playback. When used in the Command Window, the filename can be specified as part of the command.

Example:
PLAYBACK DATA1

\* NOTE

When PLAYBACK is used in a test script, the filename must be specified with =TITLE.

=TITLE ( filename -- )

Specifies the name of the file to be opened for disk recording or disk playback.

Example:
Obtain playback data from disk.
FROM_DISK            ( Identifies disk file as data source )
HALT                 ( Place the monitor in playback mode )
"DATA3" =TITLE       ( Create title for next data file to be opened )
PLAYBACK             ( Playback data )

Playback Control

The following commands control display scrolling.

FORWARD or F ( -- )
  Scrolls one line forward on the screen.

DOWN (Down arrow)

BACKWARD or B ( -- )
  Scrolls one line backward on the screen.

UP (Up arrow)

SCRN_FWD or FF ( -- )
  Scrolls one page forward on the screen.

CTRL DOWN

SCRN_BACK or BB ( -- )
  Scrolls one page backward on the screen.

CTRL UP

TOP ( -- )
  Positions the display at the beginning of the playback source.

CTRL SHIFT UP
BOTTOM ( -- )
Positions the display at the end of the playback source.

CTRL SHIFT ↓

2.3 Simultaneous Live Data and Playback

Live data can be recorded to disk while playing back data from capture RAM.

![ISDN Monitor Data Flow Diagram - Freeze Mode](image)

FROM_CAPT FREEZE
Capture topic
Record to Disk function key
Display topic
Playback RAM function key

FREEZE ( -- )
Enables data to be recorded to disk while data from capture RAM is played back.
ISDN D–Channel data can be monitored on a WAN, Basic Rate, or Primary Rate interface. This section describes configuration commands for layers 1, 2, and 3.

3.1 Layer 1

This section describes commands to configure the physical layer for each interface type.

Basic Rate

The Basic Rate interface is configured on the Home processor prior to loading the D–Channel Monitor application. However, commands exist within the application for interface configuration.

Each application processor is associated with a serial port which has a port identifier. This port identifier is used in most of the Basic Rate layer 1 configuration commands.

**PORT_BRID ( -- port id )**

Returns the port identifier for the Basic Rate D–Channel port (Port A for BRA/BRA tester).

**PORT_BRID2 ( -- port id )**

Returns the port identifier for the Basic Rate D–Channel Port B (BRA/BRA tester).

The S/T bus can be configured for point–to–point operation (one TE device attached) or point to multipoint operation (up to eight TE devices attached).

**PPMP ( port\connect mode -- )**

Where: connect mode = P2P point–to–point mode (default)

P2MP point to multipoint

Establishes the bus configuration of the unit.

Example 1:
Configure the S/T bus for point–to–point operation.

**PORT_BRID P2P PPMP**

Example 2:
Configure the S/T bus for Port B of a BRA/BRA tester for point to multipoint operation.

**PORT_BRID2 P2MP PPMP**
When a phone is connected to the tester, the voice encoding method is either A-law or μ-law.

**CODE_TYPE** ( port\voice code -- )  
Where: voice code =U-LAW μ-law voice encoding (default)  
A-LAW A-law voice encoding  

Selects the type of voice encoding.

Example:  
Select μ-law voice encoding.  
PORT_BRID U-LAW CODE_TYPE

When no data is displayed or passed to the test manager, the S/T bus could be inactive (appropriate LED's all red). The state of the S/T bus can be determined by the following three commands:

**ACTIVATED?** ( -- flag )  
Returns true if the bus is activated.

**DEACTIVATED?** ( -- flag )  
Returns true if the bus is deactivated.

**INTERRUPTED?** ( -- flag )  
Returns true if the bus is interrupted (lost framing).

Each B-Channel data stream is always routed to an application processor and the associated external connector for monitoring. To monitor data, one B-Channel can be routed to CODEC.

**BCHAN_SRC** ( port\source\Bchan -- )  
Where: source = VOICE (CODEC)  
PP1 (AP #1 / AP #4 for Port B of BRA/BRA)  
PP2 (AP #2 / AP #5 for Port B of BRA/BRA)  
Bchan = BCHAN1 (B1-Channel)  
BCHAN2 (B2-Channel)

Selects the source of monitored data for the specified B-Channel. By default, AP #1 (AP #4 for Port B of BRA/BRA) monitors B1-Channel data. AP #2 (AP #5 for Port B of BRA/BRA) monitors B2-Channel data. Data routed to AP #1 (AP #4) is also routed to the external connector #1 and data routed to AP #2 (AP #5) is also routed to the external connector #2.

Example 1:  
Route the B1-Channel data to AP #2.  
PORT_BRID PP2 BCHAN1 BCHAN_SRC

Example 2:  
Route the B2-Channel data to the voice CODEC.  
PORT_BRID VOICE BCHAN2 BCHAN_SRC

Example 3:  
Route the B2-Channel data on Port B of a BRA/BRA tester to AP #4.  
PORT_BRID2 PP1 BCHAN2 BCHAN_SRC
By default, B–Channels are connected to application processors and external connectors. The data flow to these connectors can be turned off/on.

**DROUTING (port\route -- flag)**

Where: route = DRT_OFF – turn off the flow of data to the application processors

DRT_SERIAL – route B–Channels to the application processors (default)

Connects the B–Channels to an application processor and returns true if successful.

Example:
Configure the tester to connect the B–Channels through to the application processors.

```plaintext
1 SEQ{
    PORT_BRID PP1 BCHAN1 BCHAN_SRC
    PORT_BRID PP2 BCHAN2 BCHAN_SRC
    PORT_BRID DRT_SERIAL DROUTING
    IF
        " B–Channels are connected to application processors."
    ELSE
        " Bus not activated. B–Channels are not connected."
    ENDIF
    W.NOTICE
}SEQ
```

B–Channel data can be connected to or disconnected from the external connector.

**EXTERNAL_OUT (port \ external connector \ state -- )**

Where: external connector = EXT_1 (B–Channel) or EXT_2 (B2–Channel)

state = YES (on)

NO (off)

Connects/disconnects the B–Channels to/from the external connector.

⚠️ **NOTE**

*If the B–Channels are swapped, EXT_1 will be associated with the B2–Channel and EXT_2 with the B1–Channel.*

Example:
Connect the B1–Channel data to the external connector #1.

```plaintext
PORT_BRID PP1 BCHAN1 BCHAN_SRC ( route B1–Channel to AP#1 )
PORT_BRID PP2 BCHAN2 BCHAN_SRC ( route B2–Channel to AP#2 )
PORT_BRID EXT_1 YES EXTERNAL_OUT ( Connect B1–Channel data )
```

---

**Primary Rate**

The Primary Rate interface is configured on the Home processor prior to loading the D–Channel Monitor application. Corresponding configuration commands do not exist in the application.
WAN

The WAN interface is configured on the application processor after loading and switching to the D-Channel Monitor.

<table>
<thead>
<tr>
<th>Monitor Configuration Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>➔ Interface Type</td>
</tr>
<tr>
<td>Bit Rate</td>
</tr>
<tr>
<td>Frame Sequence Number Modulo</td>
</tr>
<tr>
<td>Packet Communication SAPI</td>
</tr>
</tbody>
</table>

Figure 3–1 Monitor Configuration Menu (WAN)

INTERFACE_WAKEUP (—)
Configures the physical interface.

➔ Interface Type
IF=V28 (—)
Selects the V.28 connector and electrically isolates the other connectors on the port.

RS232C/V.28 function key

IF=V11 (—)
Selects the V.11 connector and electrically isolates the other connectors on the port.

RS422/V.11 function key

IF=V35 (—)
Selects the V.35 (default) connector and electrically isolates the other connectors on the port.

V.35 function key

IF=V36 (—)
Selects the V.36 connector and electrically isolates the other connectors on the port.

RS449/V.36 function key

NOTE
A WAN tester has a V.28, V.11, and either a V.35 or V.36 connector. These commands are only applicable if the program is running on a WAN interface.
The interface speed is measured, in bits per second, directly from the physical line.

INTERFACE-SPEED ( -- address )
Contains the current bit rate (default value is 64000) and is used by the monitor to calculate throughput measurements.

3.2 Layer 2
This section describes the various commands and variables used for layer 2 configuration. Layer 2 configuration is identical for all interface types.

Figure 3–2 Monitor Configuration Menu (Basic Rate)

→ Frame Sequence Number Modulo
L2=MOD128 ( -- )
Selects modulo 128 (default) method of decoding.

EXTENDED function key

L2=MOD8 ( -- )
Selects modulo 8 method of decoding.

NORMAL function key

MMODE–FLAG* ( -- address )
Contains a value identifying modulo 8 (0) or modulo 128 (1) method of decoding.

→ Packet Communication SAPI
Layer 3 information within layer 2 I or UI frames with a specified SAPI value is decoded according to X.25.

NOTE
This SAPI value cannot be changed within test scripts. Use the Monitor Configuration Menu to modify this SAPI.

DATA–SAPI# ( -- value )
Returns the current value of the data communications (packet data) SAPI (default value is 16).
This section describes the data flow diagram for capture to RAM and lists the commands available for test scripts. Data stored in either capture RAM or disk can be played back as described in Section 2.2. Data stored in capture RAM can be transferred to a disk file.

Figure 4-1 ISDN Data Flow Diagram - Capture to RAM

### 4.1 Capturing to RAM

**CAPT_ON ( -- )**
Saves live data in capture RAM (default).

- **Capture topic**
  - "Capture to RAM" function key (highlighted)

**CAPT_OFF ( -- )**
Live data is not saved in capture RAM.

- **Capture topic**
  - "Capture to RAM" function key (not highlighted)

**CAPT_WRAP ( -- )**
Initializes capture RAM so that new data overwrites (default) old data after the capture buffer is filled (endless loop recording).

- **Capture topic**
  - "Recording Menu"
  - "When Buffer Full"
  - "WRAP" function key
CAPT_FULL ( -- )
Initializes capture RAM so that capturing stops when the capture buffer is full.

(!(Capture topic
Recording Menu
→ When Buffer Full
STOP function key

✓ WARNING
CAPT_WRAP and CAPT_FULL erase all data in the capture RAM.

CLEAR_CAPT ( -- )
Erases all data currently in capture RAM.

(!(Capture topic
Clear function key

4.2 Transferring from RAM

Data can be transferred from capture RAM to disk, and printed as it is played back. To transfer data to disk, a data recording must be opened using the RECORD and CTOD_ON commands prior to using TRANSFER. To transfer data from capture RAM to the printer, the PRINT_ON command must first be issued. The data being transferred is displayed on the screen.

TRANSFER ( -- )
Transfers data from the selected data source.

(!(Capture topic
Save RAM to Disk function key (highlighted)

QUIT_TRA ( -- )
Abruptly terminates the transfer of data from capture RAM to disk.

(!(Capture topic
Save RAM to Disk function key (not highlighted)

TRA_ALL ( -- )
Transfers the entire contents of capture RAM (default) when the TRANSFER command is used.

(!(Capture topic
Save RAM to Disk function key
All function key
TRA_START ( -- )
Selects the starting block for transfer and is used with TRA_END when a partial transfer is desired. Use the cursor keys to locate the desired starting block prior to calling TRA_START. TRA_START selects the last scrolled block as the initial starting block for transfer.

Capture topic
- Save RAM to Disk function key
- Set Start function key

TRA_END ( -- )
Selects the final block for transfer and is used with TRA_START when a partial transfer is desired. Use the cursor keys to locate the desired final block prior to calling TRA_END. TRA_END selects the last scrolled block as the final starting block for transfer.

Capture topic
- Save RAM to Disk function key
- Set End function key

SEE_TRA ( -- )
Displays the timestamps for the initial and final blocks selected for transfer in the Command and Test Script Windows.

Example:
Open a data file with the filename ‘DATA1’ and transfer all data from capture RAM to disk. After the transfer is complete, turn off data recording.

FROM_CAPT ( Designate capture RAM as data source )
HALT ( Enter playback mode )
" DATA1" =TITLE ( Assign filename DATA1 )
RECORD ( Open data recording )
CTOD_ON ( Enable capture transfer to disk )
TRA_ALL ( Transfer all data )
TRANSFER ( Transfer data from capture to disk )
DISK_OFF ( Turn off data recording )

---

To Disk

CTOD_ON ( -- )
Enables transfer of data from capture RAM to disk when data source is playback RAM and a data recording file is open.

CTOD_OFF ( -- )
Disables transfer of data from capture RAM to disk (default) when data source is playback RAM.
To Printer

PRINT_ON ( -- )
Prints data lines as displayed during playback from either capture RAM or disk. No printout is made when the source is live data. The printer must be configured from the Printer Port Setup Menu under the Setup topic on the Home processor.

Print topic
Print On function key

PRINT_OFF ( -- )
Data is not printed during playback (default).

Print topic
Print Off function key

Example:
Transfer all data from capture RAM to the printer.

FROM_CAPT
HALT
PRINT_ON
TRA_ALL
TRANSFER

( Designate capture RAM as data source )
( Enter playback mode )
( Enable printing )
( Transfer all )
( Transfer data to printer )
Live data from the interface can be recorded to either a floppy or hard disk. Data stored in either capture RAM or disk can be played back as described in Section 2.2. Data stored in capture RAM can be transferred to disk as described in Section 4.2.

**Figure 5-1** ISDN Data Flow Diagram – Recording to Disk

**DISK_WRAP (---)**
Selects disk recording overwrite (default).

⚠️ Capture topic
Recording Menu
→ When File Full
   WRAP function key

**DISK_FULL (---)**
Turns off disk recording overwrite. Recording continues until the data recording file is full.

⚠️ Capture topic
Recording Menu
→ When File Full
   STOP function key

⚠️ WARNING
**DISK_WRAP** and **DISK_FULL** must be called prior to opening a recording with the **RECORD** command. If called while recording is in process, the status of the disk recording overwrite for this recording session will not change.
RECORD ( -- )
Opens a data recording file. When used in the Command Window, the filename can be specified as part of the command.

Example:
RECORD DATA1

Capture topic
Record to Disk function key (highlighted)

NOTE
When RECORD is used in a test script, the filename must be specified with =TITLE. Because of the relatively long time required to open a disk file (especially on a floppy drive), RECORD should not be used within time critical portions of a test script.

Trace report lines are included in the data file when an application requests start and end recording. The information in these traces identifies the traffic type and application program used while the data was being recorded.

Example:
Recording Start : ISDN Emul USR D Channel
V2.0-2.0 Rev 0 PT500 - 24 SN# 01-261
Recording End : ISDN Emul USR D Channel
V2.0-2.0 Rev 0 PT500 - 24 SN# 01-261

DISK_OFF ( -- )
Live data is not recorded to disk. The current disk recording is closed.

Capture topic
Record to Disk function key (not highlighted)

NOTE
Refer to the Programmer's Reference Manual for multi-processor disk recording.

DIS_REC ( -- )
Momentarily suspends disk recording. The data recording file remains open but no data is saved to disk.

Capture topic
Record to Disk function key (highlighted)
Suspend Recording function key (highlighted)

ENB_REC ( -- )
Enables data recording. The data recording file remains open and live data is recorded to disk.

Capture topic
Record to Disk function key (highlighted)
Suspend Recording function key (not highlighted)
The ISDN D-Channel Monitor and Emulation applications can display data from the line (live data), from capture RAM, or from a disk recording in a variety of formats.

The data flow diagram for displaying and printing data, as well as commands available for test scripts, are described in this section.

![ISDN Data Flow Diagram - Display and Print](image)

**Figure 6-1. ISDN Data Flow Diagram – Display and Print**

⚠️ **NOTE**

Data can only be printed in playback mode.
The default display is normal format. Timestamp, Layer 1 Report, Layer 2 Report, and Layer 3 Report can only be modified when Display Format is set to NORMAL.

REP_NORMAL (---)
Turns on data display (default).

Format topic
NORMAL function key

REP_OFF (---)
Turns off data display.

OFF function key

REP_SPLIT (---)
Displays data with a split screen display. The screen is divided in half vertically, with frames sent from the network displayed on the left and frames sent from the user on the right.

Format topic
SPLIT function key

NOTE
Only the first 38 characters of a trace statement are displayed when split display format is selected.

REP_TRACE (---)
Displays only trace statements.

TRACE function key
→ **Timestamp**

**TIME_OFF ( -- )**
Timestamps are not displayed (default). Block sequence numbers for each received frame or physical event are displayed.

achment OFF function key

**TIME_ON ( -- )**
Displays the start and end of timestamps as minutes, seconds, and tenths of milliseconds. Block sequence numbers for received frames or physical events are not displayed.

achment MM:SS.ssss function key

**TIME_DAY ( -- )**
Displays the start and end of timestamps as days, hours, minutes, and seconds. Block sequence numbers for received frames or physical events are not displayed.

achment DD HH:MM:SS function key

**BLOCK_ON ( -- )**
Displays block sequence numbers for each received frame or physical event. BLOCK_ON is functionally identical to TIME_OFF.

achment OFF function key

→ **Layer 1 Report**
Reports physical interface (layer 1) events to the data display as they occur.

**L1_OFF ( -- )**
Layer 1 events are not displayed.

achment OFF function key

**L1_COMP ( -- )**
Displays layer 1 events in complete format (default).

achment ON function key

→ **Layer 2 Report**
Reports received frames in a variety of display formats.

**FRM_OFF ( -- )**
Received frames are not displayed.

achment OFF function key
FRM_COMP ( -- )
Displays received frames in complete format. The SAPI, TEI, C/R bit, P/F bit, and frame type of each received frame is displayed. The NR and NS values are displayed for I frames. Management UI frames, FRMR, and XID frames are also completely decoded and displayed.

L.2 Comp function key

FRM_MNEM ( -- )
Displays received frames in mnemonic format (default). Only the SAPI, TEI, C/R bit, and frame type of each received frame is displayed.

L.2 Mnem function key

FRM_HEX ( -- )
Displays received frames in hexadecimal format. Each byte of the layer 2 frame is displayed as two hexadecimal digits.

HEX function key

FRM_CHAR ( -- )
Displays received frames in the currently selected character set.

TEXT function key

Layer 3 Report
Reports the information fields of received layer 2 I frames as layer 3 messages in a variety of display formats.

MSG_OFF ( -- )
Layer 3 messages are not displayed.

OFF function key

MSG_COMP ( -- )
Displays layer 3 messages in complete format. The layer 3 message is decoded as completely as possible. The protocol discriminator, call reference (value and flag), and message type fields are displayed, followed by a line–by–line listing of each parameter of each information element in the message. The value and meaning of each parameter is displayed.

L.3 Comp function key

NOTE
Because of the amount of detail this display outputs, it is not recommended for displaying live data.
**MSG_MNEM** (**)  
Displays layer 3 messages in mnemonic format (default). The protocol discriminator, call reference (value and flag), and message type fields are displayed. The message type is displayed as a mnemonic name. For example, the ‘CALL PROCeeding’ message is displayed as ‘CALLPROC’.

![L.3 Mnem function key]

**MSG_HEX** (**)  
Displays layer 3 messages in hexadecimal format. Each byte of the layer 3 message is displayed as two hexadecimal digits.

![HEX function key]

**MSG_CHAR** (**)  
Displays layer 3 messages in the currently selected character set.

![TEXT function key]

> Message Detail  
Selects the amount of detail for a layer 3 message displayed in mnemonic, text, or hexadecimal format.

**MSG** (**)  
Selects only the message header for display. Decodes and displays the protocol discriminator, call reference (value and flag), and message type fields.

![MSG function key]

**MSG+IE** (**)  
Selects the message header and information element headers for display. Decodes and displays the protocol discriminator, call reference (value and flag), message type fields, and the information element code for each IE in the message.

![MSG+IE function key]

**MSG+IE+PA** (**)  
Selects the message header, information element headers, and information element parameter fields for display. Decodes and displays the protocol discriminator, call reference (value and flag), message type fields, and the parameter fields of each IE in the message.

![MSG+IE+PA function key]
Packet Data
Selects the display format for X.25 data contained in a SAPI 16 I frame.

DATA_OFF ( -- )
Packet data is not displayed.

OFF function key

DATA_HEX ( -- )
Displays packet data in hexadecimal format.

HEX function key

DATA_CHAR ( -- )
Displays packet data in the currently selected character set.

CHAR function key

Character Set
Selects the character set for layer 2 or layer 3 data display.

CS=ASCII ( -- )
Sets the character set for data display to ASCII (default).

ASCII function key

CS=EBCDIC ( -- )
Sets the character set for data display to EBCDIC.

EBCDIC function key

CS=HEX ( -- )
Sets the character set for data display to hex.

HEX function key

CS=JIS8 ( -- )
Sets the character set for data display to JIS8.

JIS8 function key
Dual Window
If two applications have been loaded, the screen can be divided horizontally to display data from both applications. The current application is always displayed in the top window.

FULL ( -- )
Uses the entire Data Display Window for the current application.

Dual window commands vary depending on the machine configuration. Table 6–1 shows the relationship between machine configuration, application processors, and dual window commands.

<table>
<thead>
<tr>
<th>Machine Type</th>
<th>Command</th>
<th>Dual Window AP #</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAN/WAN</td>
<td>DUAL_1+2</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+2</td>
<td>AP #2</td>
</tr>
<tr>
<td>BRA/WAN</td>
<td>DUAL_1+7</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+7</td>
<td>AP #2</td>
</tr>
<tr>
<td>PRA</td>
<td>DUAL_3+4</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AP #2</td>
</tr>
<tr>
<td>PRA/BRA/WAN</td>
<td>DUAL_1+2</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+3</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+4</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+7</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+3</td>
<td>AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+4</td>
<td>AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+7</td>
<td>AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+4</td>
<td>AP #4</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+7</td>
<td>AP #4</td>
</tr>
<tr>
<td></td>
<td>DUAL_4+7</td>
<td>AP #5</td>
</tr>
<tr>
<td>BRA/BRA</td>
<td>DUAL_1+2</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+3</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+4</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+5</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+7</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+3</td>
<td>AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+4</td>
<td>AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+5</td>
<td>AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+7</td>
<td>AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+4</td>
<td>AP #4</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+5</td>
<td>AP #4</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+7</td>
<td>AP #4</td>
</tr>
<tr>
<td></td>
<td>DUAL_4+5</td>
<td>AP #5</td>
</tr>
<tr>
<td></td>
<td>DUAL_4+7</td>
<td>AP #5</td>
</tr>
<tr>
<td></td>
<td>DUAL_5+7</td>
<td>AP #6</td>
</tr>
<tr>
<td>PRA/WAN</td>
<td>DUAL_1+3</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+4</td>
<td>AP #1</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+4</td>
<td>AP #2</td>
</tr>
</tbody>
</table>

Table 6–1 Dual Window Commands
Trace Display Format
Selects the display format for trace statements.

TRACE_SHORT ( -- )
Displays the trace statement on one line (short format) containing only user-defined text.

SHORT function key

TRACE_COMP ( -- )
Displays the trace statement on two lines (complete format). Block sequence numbers or timestamps are displayed on the first line, and user-defined text on the second line.

COMPLETE function key

Throughput Graph
The throughput rate can be calculated, displayed as a bar graph, and printed out. The ISDN Monitor calculates throughput by counting the number of bytes on each side of the line during two intervals – one short, one long. This figure is divided by the time interval to arrive at a bits per second figure for each time interval (for both user and network data).

NOTE
For accurate throughput measurement, the bit rate (line speed) must be set to match the actual line speed. For WAN applications, set the speed on the Monitor/Emulation Configuration Menu.

INTERFACE-SPEED ( -- address )
Contains the current bit rate (default value is 16000 for Basic Rate applications, and 64000 for Primary Rate and WAN applications).

Example:
In the WAN application, set the throughput measurement speed to 2400.
2400 INTERFACE-SPEED !
TPR_ON

TPR_ON ( -- )
Calculates and displays the throughput rate as a bar graph.

DISPLAY function key

WARNING
If the short interval, long interval, or speed is changed, TPR_ON must be called after the changes are made.

TPR_OFF ( -- )
The throughput rate is not calculated or displayed.

OFF function key
PRINT_TPR ( -- )
Calculates and displays the throughput rate as a bar graph, and prints the long term interval measurements.

DISPLAY AND PRINT function key

⇒ Short Interval (sec)
Sets the short time interval, in seconds, for measuring, displaying, and printing the throughput results.

SHORT-INTERVAL ( -- address )
Contains the current duration of the short interval (default value is 10 seconds).

Example:
Set the short interval to 20 seconds.
20 SHORT-INTERVAL !
TPR_ON

Modify Short Interval function key

⇒ Long Interval (sec)
Sets the long time interval, in seconds, for measuring, displaying, and printing the throughput results.

LONG-INTERVAL ( -- address )
Contains the current duration of the long interval (default value is 600 seconds).

Example:
Set the long interval to 300 seconds.
300 LONG-INTERVAL !
TPR_ON

Modify Long Interval function key
Filters provide the capability of passing or blocking specific events from the display, capture RAM, or disk recording. These three filters act independently. This section describes the commands used to pass or block individual events, and activate or deactivate each of the three filters.

### 7.1 Layer 1

#### Figure 7-1 Layer 1 Filter Setup Menu (Basic Rate)

- **Filter Type**
  - RAM
  
- **Filter Status** DEACTIVATED

<table>
<thead>
<tr>
<th>Layer 1 Events:</th>
<th>ACTIVATE</th>
<th>PASS</th>
<th>Ri2Err</th>
<th>PASS</th>
<th>Recovery</th>
<th>PASS</th>
<th>Undefined</th>
<th>PASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVATE</td>
<td>PASS</td>
<td></td>
<td>Ri2Err</td>
<td>PASS</td>
<td>Recovery</td>
<td>PASS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEACTIVATE</td>
<td>PASS</td>
<td></td>
<td>LOST FRAMING</td>
<td>PASS</td>
<td>UNDEFINED</td>
<td>PASS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Filter Type**
  - There are three separate filter processes which act independently of each other: DISPLAY, RAM, and DISK.

- **Filter Status**
  - Filters can be deactivated (default) or activated at any time.

When a filter is deactivated, the pass or block settings for individual events are ignored so that all events are passed. When a filter is activated, any previous settings are in effect.

**ACTIVATE_REPORT ( — )**
- Activates the display (report) filter.

- **Filters topic**
  - Activate Display Filter function key (highlighted).
DEACTIVATE_REPORT (--)
Deactivates the display filter.

Filters topic
Activate Display Filter function key (not highlighted)

ACTIVATE_RAM (--)
Activates the capture RAM filter.

Filters topic
Activate RAM Filter function key (highlighted)

DEACTIVATE_RAM (--)
Deactivates the capture RAM filter.

Filters topic
Activate RAM Filter function key (not highlighted)

ACTIVATE_DISK (--)
Activates the disk filter.

Filters topic
Activate Disk Filter function key (highlighted)

DEACTIVATE_DISK (--)
Deactivates the disk filter.

Filters topic
Activate Disk Filter function key (not highlighted)

Layer 1 Events:
Physical (layer 1) events can be blocked or passed (default).

NOTE
Commands to change filter status are prefixed with 'R' for display filters, 'C' for capture RAM filters, and 'D' for disk recording filters.

R1=None (--)
Blocks layer 1 events from the display.

R1=All (--)
Passes layer 1 events to the display.

C1=None (--)
Blocks layer 1 events from capture RAM.

C1=All (--)
Passes layer 1 events to capture RAM.
**D1=NONE** (---)
Blocks layer 1 events from disk.

**D1=ALL** (---)
Passes layer 1 events to disk.

Table 7-1 summarizes the layer 1 filter commands for the Basic Rate interface. Table 7-2 summarizes the layer 1 filter commands for the Primary Rate interface. There are no layer 1 filter commands for the WAN interface.

<table>
<thead>
<tr>
<th>Description</th>
<th>Display</th>
<th>RAM</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activated</td>
<td>Pass R1+ACTIVATED</td>
<td>C1+ACTIVATED</td>
<td>D1+ACTIVATED</td>
</tr>
<tr>
<td></td>
<td>Block R1-ACTIVATED</td>
<td>C1-ACTIVATED</td>
<td>D1-ACTIVATED</td>
</tr>
<tr>
<td>Deactivated</td>
<td>Pass R1+DEACTIVATED</td>
<td>C1+DEACTIVATED</td>
<td>D1+DEACTIVATED</td>
</tr>
<tr>
<td></td>
<td>Block R1-DEACTIVATED</td>
<td>C1-DEACTIVATED</td>
<td>D1-DEACTIVATED</td>
</tr>
<tr>
<td>Lost Framing</td>
<td>Pass R1+LSTFRM</td>
<td>C1+LSTFRM</td>
<td>D1+LSTFRM</td>
</tr>
<tr>
<td></td>
<td>Block R1-LSTFRM</td>
<td>C1-LSTFRM</td>
<td>D1-LSTFRM</td>
</tr>
<tr>
<td>Recovery</td>
<td>Pass R1+RECOVERY</td>
<td>C1+RECOVERY</td>
<td>D1+RECOVERY</td>
</tr>
<tr>
<td></td>
<td>Block R1-RECOVERY</td>
<td>C1-RECOVERY</td>
<td>D1-RECOVERY</td>
</tr>
<tr>
<td>INFO 2 Error</td>
<td>Pass R1+RI2ERR</td>
<td>C1+RI2ERR</td>
<td>D1+RI2ERR</td>
</tr>
<tr>
<td></td>
<td>Block R1-RI2ERR</td>
<td>C1-RI2ERR</td>
<td>D1-RI2ERR</td>
</tr>
<tr>
<td>Undefined</td>
<td>Pass R1+UNDEFINED</td>
<td>C1+UNDEFINED</td>
<td>D1+UNDEFINED</td>
</tr>
<tr>
<td></td>
<td>Block R1-UNDEFINED</td>
<td>C1-UNDEFINED</td>
<td>D1-UNDEFINED</td>
</tr>
</tbody>
</table>

**Table 7-1 Basic Rate Layer 1 Filter Commands**

**Figure 7-2 Layer 1 Filter Setup Menu (Primary Rate)**
### Table 7–2 Primary Rate Layer 1 Filter Commands

<table>
<thead>
<tr>
<th>Description</th>
<th>Display</th>
<th>RAM</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Alarm</td>
<td>Pass</td>
<td>R1+RED_ALARM</td>
<td>C1+RED_ALARM</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R1–RED_ALARM</td>
<td>C1–RED_ALARM</td>
</tr>
<tr>
<td>Yellow Alarm</td>
<td>Pass</td>
<td>R1+YEL_ALARM</td>
<td>C1+YEL_ALARM</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R1–YEL_ALARM</td>
<td>C1–YEL_ALARM</td>
</tr>
<tr>
<td>Lost Signal</td>
<td>Pass</td>
<td>R1+LST_SIGL</td>
<td>C1+LST_SIGL</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R1–LST_SIGL</td>
<td>C1–LST_SIGL</td>
</tr>
<tr>
<td>Lost Phase</td>
<td>Pass</td>
<td>R1+LST_PLOCK</td>
<td>C1+LST_PLOCK</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R1–LST_PLOCK</td>
<td>C1–LST_PLOCK</td>
</tr>
<tr>
<td>Synchronized</td>
<td>Pass</td>
<td>R1+NORMAL</td>
<td>C1+NORMAL</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R1–NORMAL</td>
<td>C1–NORMAL</td>
</tr>
<tr>
<td>Out of Frame</td>
<td>Pass</td>
<td>R1+OUT_OF_FRM</td>
<td>C1+OUT_OF_FRM</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R1–OUT_OF_FRM</td>
<td>C1–OUT_OF_FRM</td>
</tr>
</tbody>
</table>
7.2 Layer 2

Layer 2 Filter Setup Menu

- Filter Type  RAM
  Filter Status  DEACTIVATED

Link Address Events:
  SAPI Filter  PASS  SAPI  0
  TEI Filter  PASS  TEI  2
  Logical Operation  OR

Layer 2 Events:
  RR  PASS  SABM  PASS  I  PASS  DISC  PASS  INVALID  PASS
  RNR  PASS  SABME  PASS  UA  PASS  DM  PASS
  REJ  PASS  XID  PASS  UI  PASS  FRMR  PASS

Figure 7–3  Layer 2 Filter Setup Menu

Link Address Events:
A filter condition can be set to pass or block on a specific SAPI or TEI value. The SAPI and TEI conditions can be logically combined to provide complex filtering capabilities. There are no commands available; use the function keys to set SAPI and TEI conditions.

Layer 2 Events:
Frame (layer 2) events can be blocked or passed (default).

R2=NONE  ( -- )
  Blocks layer 2 events from the display.

R2=ALL  ( -- )
  Passes layer 2 events to the display.

C2=NONE  ( -- )
  Blocks layer 2 events from capture RAM.

C2=ALL  ( -- )
  Passes layer 2 events to capture RAM.

D2=NONE  ( -- )
  Blocks layer 2 events from disk.

D2=ALL  ( -- )
  Passes layer 2 event reports to disk.
Table 7-3 summarizes the layer 2 filter commands.

<table>
<thead>
<tr>
<th>Description</th>
<th>Display</th>
<th>RAM</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Asynchronous Balanced Mode</td>
<td>Pass</td>
<td>R2+SABM</td>
<td>C2+SABM D2+SABM</td>
</tr>
<tr>
<td>Block</td>
<td>R2−SABM</td>
<td>C2−SABM</td>
<td>D2−SABM</td>
</tr>
<tr>
<td>Set Asynchronous Balance Mode Extended</td>
<td>Pass</td>
<td>R2+SABME</td>
<td>C2+SABME D2+SABME</td>
</tr>
<tr>
<td>Block</td>
<td>R2−SABME</td>
<td>C2−SABME</td>
<td>D2−SABME</td>
</tr>
<tr>
<td>Unnumbered Acknowledgement</td>
<td>Pass</td>
<td>R2+UA</td>
<td>C2+UA D2+UA</td>
</tr>
<tr>
<td>Block</td>
<td>R2−UA</td>
<td>C2−UA</td>
<td>D2−UA</td>
</tr>
<tr>
<td>Information</td>
<td>Pass</td>
<td>R2+I</td>
<td>C2+I D2+I</td>
</tr>
<tr>
<td>Block</td>
<td>R2−I</td>
<td>C2−I</td>
<td>D2−I</td>
</tr>
<tr>
<td>Disconnect Mode</td>
<td>Pass</td>
<td>R2+DISC</td>
<td>C2+DISC D2+DISC</td>
</tr>
<tr>
<td>Block</td>
<td>R2−DISC</td>
<td>C2−DISC</td>
<td>D2−DISC</td>
</tr>
<tr>
<td>Reject</td>
<td>Pass</td>
<td>R2+REJ</td>
<td>C2+REJ D2+REJ</td>
</tr>
<tr>
<td>Block</td>
<td>R2−REJ</td>
<td>C2−REJ</td>
<td>D2−REJ</td>
</tr>
<tr>
<td>Receive Not Ready</td>
<td>Pass</td>
<td>R2+RNR</td>
<td>C2+RNR D2+RNR</td>
</tr>
<tr>
<td>Block</td>
<td>R2−RNR</td>
<td>C2−RNR</td>
<td>D2−RNR</td>
</tr>
<tr>
<td>Receive Ready</td>
<td>Pass</td>
<td>R2+RR</td>
<td>C2+RR D2+RR</td>
</tr>
<tr>
<td>Block</td>
<td>R2−RR</td>
<td>C2−RR</td>
<td>D2−RR</td>
</tr>
<tr>
<td>Frame Reject</td>
<td>Pass</td>
<td>R2+FRMR</td>
<td>C2+FRMR D2+FRMR</td>
</tr>
<tr>
<td>Block</td>
<td>R2−FRMR</td>
<td>C2−FRMR</td>
<td>D2−FRMR</td>
</tr>
<tr>
<td>Disconnect Mode</td>
<td>Pass</td>
<td>R2+DM</td>
<td>C2+DM D2+DM</td>
</tr>
<tr>
<td>Block</td>
<td>R2−DM</td>
<td>C2−DM</td>
<td>D2−DM</td>
</tr>
<tr>
<td>Unnumbered Information</td>
<td>Pass</td>
<td>R2+UI</td>
<td>C2+UI D2+UI</td>
</tr>
<tr>
<td>Block</td>
<td>R2−UI</td>
<td>C2−UI</td>
<td>D2−UI</td>
</tr>
<tr>
<td>Exchange Identification</td>
<td>Pass</td>
<td>R2+XID</td>
<td>C2+XID D2+XID</td>
</tr>
<tr>
<td>Block</td>
<td>R2−XID</td>
<td>C2−XID</td>
<td>D2−XID</td>
</tr>
<tr>
<td>Invalid</td>
<td>Pass</td>
<td>R2+INV</td>
<td>C2+INV D2+INV</td>
</tr>
<tr>
<td>Block</td>
<td>R2−INV</td>
<td>C2−INV</td>
<td>D2−INV</td>
</tr>
</tbody>
</table>

Table 7-3 Layer 2 Filter Commands
7.3 Layer 3

**Layer 3 Filter Setup Menu**

<table>
<thead>
<tr>
<th>Filter Type</th>
<th>Filter Status</th>
<th>Protocol Discriminator</th>
<th>Message Set</th>
<th>PD Value</th>
<th>CR Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALERT</td>
<td>BLK</td>
<td>REL</td>
<td>BLK</td>
<td>SUSP</td>
<td>BLK</td>
</tr>
<tr>
<td>CALL_PROC</td>
<td>BLK</td>
<td>REL_COM</td>
<td>BLK</td>
<td>SUSP_ACK</td>
<td>BLK</td>
</tr>
<tr>
<td>CON_CONN</td>
<td>BLK</td>
<td>RES</td>
<td>BLK</td>
<td>SUSP_REJ</td>
<td>BLK</td>
</tr>
<tr>
<td>CONN_ACK</td>
<td>BLK</td>
<td>RES_ACK</td>
<td>BLK</td>
<td>USER_INFO</td>
<td>BLK</td>
</tr>
<tr>
<td>DISC</td>
<td>BLK</td>
<td>SETUP</td>
<td>PASS</td>
<td>REST</td>
<td>BLK</td>
</tr>
<tr>
<td>INFO</td>
<td>PASS</td>
<td>SETUP_ACK</td>
<td>BLK</td>
<td>SEGMENT</td>
<td>BLK</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>BLK</td>
<td>STATUS</td>
<td>BLK</td>
<td>FAC</td>
<td>BLK</td>
</tr>
<tr>
<td>PROG</td>
<td>BLK</td>
<td>STATUS_EN</td>
<td>BLK</td>
<td>HOLD</td>
<td>BLK</td>
</tr>
</tbody>
</table>

**Figure 7-4 Layer 3 Filter Setup Menu**

→ Protocol Discriminator
The protocol discriminator of layer 3 messages can be filtered. The type of filter must be selected first.

**F-PD (--)**
Ignores the message protocol discriminator when filtering.

✓ OFF function key

**F+PD (--)**
Passes data with the specified protocol discriminator.

✓ PASS function key

**=F_PD (value --)**
Where: value = protocol discriminator

Specifies the protocol discriminator value. Only messages with this protocol discriminator value are passed if the protocol discriminator filter is on.
→ Call Reference
The call reference fields of layer 3 messages can be filtered. The type of filter must be selected first.

F-CR ( --- )
   Ignores the message call reference when filtering.

   OFF function key

F+CR ( --- )
   Passes data with the specified call reference.

   PASS function key

=F.CR ( value -- )
   Where: value = call reference value

   Specifies the call reference value. Only messages with this call reference value are passed if the call reference filter is on.

Layer 3 Messages
Message (layer 3) events can be blocked or passed (default). The layer 2 I frame filter must be set to pass for the layer 3 filters to have any effect.

The commands to pass or block individual messages differ from the layer 1 and layer 2 filter commands. The type of filter (display, capture RAM, or disk) must be selected first. Individual messages can then be set to pass or block for each filter.

R_FILTER ( -- )
   Selects the display (report) filter.

C_FILTER ( -- )
   Selects the capture RAM filter.

D_FILTER ( -- )
   Selects the disk recording filter.

F3=NONE ( -- )
   Blocks all layer 3 messages from passing through the selected filter.

F3=ALL ( -- )
   Passes all layer 3 messages through the selected filter.

F-MSG ( message identifier -- )
   Blocks the specified message from passing through the selected filter. Valid message identifiers are listed in each message set manual.

F+MSG ( message identifier -- )
   Passes the specified message through the selected filter.
Example:
Set the display filter to pass SETUP messages only (CCITT message set).

R_FILTER
F3=NONE
M#SETUP F+MSG

\[ NOTE \]
The special message identifiers M#INVALID and M#UNDEF may be used to filter invalid or undefined messages respectively.

### 7.4 X.25 Layer 3

<table>
<thead>
<tr>
<th>Filter Type</th>
<th>Display</th>
<th>Selective Address</th>
<th>Selective LCN #1</th>
<th>Selective LCN #2</th>
<th>Selective LCN #3</th>
<th>Selective LCN #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter Status</td>
<td>DEACTIVATED</td>
<td>Selective LCN #1</td>
<td>ALL</td>
<td>LCN #2</td>
<td>ALL</td>
<td>LCN #4</td>
</tr>
<tr>
<td>Packet Layer:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call</td>
<td>PASS</td>
<td>RR</td>
<td>PASS</td>
<td>Restart</td>
<td>PASS</td>
<td>Registration</td>
</tr>
<tr>
<td>Clear</td>
<td>PASS</td>
<td>RNR</td>
<td>PASS</td>
<td>Reset</td>
<td>PASS</td>
<td>Diagnostic</td>
</tr>
<tr>
<td>Data</td>
<td>PASS</td>
<td>REJ</td>
<td>PASS</td>
<td>Interrupt</td>
<td>PASS</td>
<td>Invalid</td>
</tr>
</tbody>
</table>

\[ Selective Address \]
If filters are activated and a specific address has been entered, no SAPI 16 I frames are displayed until a call request/incoming call packet with that called or calling address is received. The call request/incoming call is shown and all subsequent SAPI 16 I frames and X.25 layer 3 packets with a PASS status, received on that LCN, are displayed until a clear request/indication packet on that LCN is received. After the clear request/indication packet, no further SAPI 16 I frames are displayed until another call request/incoming call containing the specified address is received.

\[ WARNING \]
If a second call request/incoming call, containing the specified called/calling address, is received on a different LCN, only traffic on the new LCN is displayed.

\[ NOTE \]
If a selective address is specified, the selective LCN's cannot be used.
RTRIG_ON (---)
Activates call address filtering for display filters. All X.25 packets in SAPI 16 I frames are blocked until a call request/incoming call packet is received containing the specified address. All packets on the same LCN are then passed until a clear request/indication packet is received on the same LCN. After this packet is passed, all packets in SAPI 16 I frames are blocked again. Signalling information in SAPI 0 I frames is not affected by this filter.

[Filter Type]
DISPLAY function key
[Selective Address]
ONE function key
[Filter Status]
ACTIVATED function key

CTRIG_ON (---)
Activates call address filtering for capture RAM filters. See description under RTRIG_ON.

[Filter Type]
RAM function key
[Selective Address]
ONE function key
[Filter Status]
ACTIVATED function key

DTRIG_ON (---)
Activates call address filtering for disk filters. See description under RTRIG_ON.

[Filter Type]
DISK function key
[Selective Address]
ONE function key
[Filter Status]
ACTIVATED function key

RTRIG_OFF (---)
Turns off call address filtering for display filters.

[Filter Type]
DISPLAY function key
[Selective Address]
ALL function key

CTRIG_OFF (---)
Turns off call address filtering for capture RAM filters.

[Filter Type]
RAM function key
[Selective Address]
ALL function key
DTRIG_OFF ( -- )
Turns off call address filtering for disk filters.

|$\rightarrow$ Filter Type
DISK function key
|$\rightarrow$ Selective Address
ALL function key

R-WCALLED ( -- address )
Contains a 16 byte string identifying the display selective address. The first byte of the string contains the length of the address.

Example:
Set the display filter selective address to 43042001.

```
"43042001"   ( Define the selective address )
COUNT 15 MIN  ( No more than 15 characters allowed )
DUP R-WCALLED C!  ( Store count at first byte )
R-WCALLED 1+ SWAP CMOVE  ( Move characters to R-WCALLED )
RTRIG_ON       ( Use Display selective address filter )
ACTIVATE_REPORT ( Activate Display filter )
```

|$\rightarrow$ Filter Type
DISPLAY function key
|$\rightarrow$ Selective Address
ONE function key
|$\rightarrow$ Filter Status
ACTIVATED function key

|$\downarrow$ NOTE
Use this variable to define a display filter selective address within a test script.

C-WCALLED ( -- address )
Contains a 16 byte string that identifying the RAM filter selective address. The first byte of the string contains the length of the address. See example under R-WCALLED.

D-WCALLED ( -- address )
Contains a 16 byte string that identifying the disk filter selective address. The first byte of the string contains the length of the address. See example under R-WCALLED.
Selective LCN

If display filters are activated and a specific LCN is entered, only packets that have a PASS status on that LCN are displayed. Up to four LCN's can be selected.

NOTE
Commands for display filters and selective LCN #1 are described here as an example. For a complete list of commands, see Table 7-4.

RLCN=ALL ( -- )  
Passes packets, on all LCN's, to the display.

FILTER Type  
DISPLAY function key  
→ Selective LCN #1  
ALL function key  

=RLCN1 ( LCN value -- )  
Specifies the logical channel for which packets are passed to the display.

Example:
Specify logical channel number 5 for Selective LCN #1.
5 =RLCN1  

FILTER Type  
DISPLAY function key  
→ Selective LCN #1  
Modify function key  

RLCN1=SEL ( -- )  
Passes packets to the display on the specified logical channel, as defined with the =RLCN1 command (default value is 0).

Example:
Set Selective LCN #1 to pass all packets on logical channel 20.
20 =RLCN1  
RLCN1=SEL  

FILTER Type  
DISPLAY function key  
→ Selective LCN #1  
Select function key  

RLCN1=OFF ( -- )  
Selective LCN #1 is not used for display filters.

FILTER Type  
DISPLAY function key  
→ Selective LCN #1  
OFF function key
### Table 7-4 Selective Filter Commands

<table>
<thead>
<tr>
<th>Description</th>
<th>Display</th>
<th>RAM</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>All LCNs passed</td>
<td>RLCN=ALL</td>
<td>CLCN=ALL</td>
<td>DLCN=ALL</td>
</tr>
<tr>
<td>Sets Selective LCN #1</td>
<td>=RLCN1</td>
<td>=CLCN1</td>
<td>=DLCN1</td>
</tr>
<tr>
<td>Uses Selective LCN #1 for filters</td>
<td>RLCN1=SEL</td>
<td>CLCN1=SEL</td>
<td>DLCN1=SEL</td>
</tr>
<tr>
<td>Selective LCN #1 not used for filters</td>
<td>RLCN1=OFF</td>
<td>CLCN1=OFF</td>
<td>DLCN1=OFF</td>
</tr>
<tr>
<td>Sets Selective LCN #2</td>
<td>=RLCN2</td>
<td>=CLCN2</td>
<td>=DLCN2</td>
</tr>
<tr>
<td>Uses Selective LCN #2 for filters</td>
<td>RLCN2=SEL</td>
<td>CLCN2=SEL</td>
<td>DLCN2=SEL</td>
</tr>
<tr>
<td>Selective LCN #2 not used for filters</td>
<td>RLCN2=OFF</td>
<td>CLCN2=OFF</td>
<td>DLCN2=OFF</td>
</tr>
<tr>
<td>Sets Selective LCN #3</td>
<td>=RLCN3</td>
<td>=CLCN3</td>
<td>=DLCN3</td>
</tr>
<tr>
<td>Uses Selective LCN #3 for filters</td>
<td>RLCN3=SEL</td>
<td>CLCN3=SEL</td>
<td>DLCN3=SEL</td>
</tr>
<tr>
<td>Selective LCN #3 not used for filters</td>
<td>RLCN3=OFF</td>
<td>CLCN3=OFF</td>
<td>DLCN3=OFF</td>
</tr>
<tr>
<td>Sets Selective LCN #4</td>
<td>=RLCN4</td>
<td>=CLCN4</td>
<td>=DLCN4</td>
</tr>
<tr>
<td>Uses Selective LCN #4 for filters</td>
<td>RLCN4=SEL</td>
<td>CLCN4=SEL</td>
<td>DLCN4=SEL</td>
</tr>
<tr>
<td>Selective LCN #4 not used for filters</td>
<td>RLCN4=OFF</td>
<td>CLCN4=OFF</td>
<td>DLCN4=OFF</td>
</tr>
</tbody>
</table>

**Packet Layer:**
If display filters are activated, any packet with a PASS status is displayed unless the selective address or a selective LCN has been set to a specific value. See the description under Selective Address and Selective LCN.

**NOTE**
If the I frame has been blocked, a filter at the packet layer is inappropriate. This is indicated by the dashes shown beside the individual packet layer items.

**NOTE**
Commands for display filters and call packets are described here as an example. For a complete list of commands, see Table 7-5.

**R3+CALL ( -- )**
Passes call request/incoming call and call connected/accepted packets to the display.

- **Filter Type**
  - DISPLAY function key
- **Call**
  - PASS function key
R3−CALL ( -- )
Blocks call request/incoming call and call connected/accepted packets from the display.

- Filter Type
  - DISPLAY function key
  - Call
  - BLOCK function key

R3=ALL ( -- )
Passes all packets (default) to the display.

- Filter Type
  - DISPLAY function key
  - Call
  - ALL PACKETS function key

R3=NONE ( -- )
Blocks all packets from the display.

- Filter Type
  - DISPLAY function key
  - Call
  - NO PACKETS function key
<table>
<thead>
<tr>
<th>Description</th>
<th>Display</th>
<th>RAM</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Pass (default)</td>
<td>R3=ALL</td>
<td>C3=ALL</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R3=None</td>
<td>C3=None</td>
</tr>
<tr>
<td>Call Request</td>
<td>Pass</td>
<td>R3+Call</td>
<td>C3+Call</td>
</tr>
<tr>
<td>Call Connect</td>
<td>Block</td>
<td>R3-Call</td>
<td>C3-Call</td>
</tr>
<tr>
<td>Clear Request</td>
<td>Pass</td>
<td>R3+Clear</td>
<td>C3+Clear</td>
</tr>
<tr>
<td>Clear Confirmation</td>
<td>Block</td>
<td>R3-Clear</td>
<td>C3-Clear</td>
</tr>
<tr>
<td>Reset Request</td>
<td>Pass</td>
<td>R3+Reset</td>
<td>C3+Reset</td>
</tr>
<tr>
<td>Reset Confirmation</td>
<td>Block</td>
<td>R3-Reset</td>
<td>C3-Reset</td>
</tr>
<tr>
<td>Restart Request</td>
<td>Pass</td>
<td>R3+Restart</td>
<td>C3+Restart</td>
</tr>
<tr>
<td>Restart Confirmation</td>
<td>Block</td>
<td>R3-Restart</td>
<td>C3-Reset</td>
</tr>
<tr>
<td>Interrupt</td>
<td>Pass</td>
<td>R3+INT</td>
<td>C3+INT</td>
</tr>
<tr>
<td>Interrupt Confirmation</td>
<td>Block</td>
<td>R3-INT</td>
<td>C3-INT</td>
</tr>
<tr>
<td>Data</td>
<td>Pass</td>
<td>R3+DATA</td>
<td>C3+DATA</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R3-DATA</td>
<td>C3-DATA</td>
</tr>
<tr>
<td>Receive Ready</td>
<td>Pass</td>
<td>R3+RR</td>
<td>C3+RR</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R3-RR</td>
<td>C3-RR</td>
</tr>
<tr>
<td>Receive Not Ready</td>
<td>Pass</td>
<td>R3+RNR</td>
<td>C3+RNR</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R3-RNR</td>
<td>C3-RNR</td>
</tr>
<tr>
<td>Reject</td>
<td>Pass</td>
<td>R3+REJ</td>
<td>C3+REJ</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R3-REJ</td>
<td>C3-REJ</td>
</tr>
<tr>
<td>Registration Request</td>
<td>Pass</td>
<td>R3+REG</td>
<td>C3+REG</td>
</tr>
<tr>
<td>Registration Confirmation</td>
<td>Block</td>
<td>R3-REG</td>
<td>C3-REG</td>
</tr>
<tr>
<td>Diagnostic</td>
<td>Pass</td>
<td>R3+DIAG</td>
<td>C3+DIAG</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R3-DIAG</td>
<td>C3-DIAG</td>
</tr>
<tr>
<td>Invalid</td>
<td>Pass</td>
<td>R3+INV</td>
<td>C3+INV</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>R3-INV</td>
<td>C3-INV</td>
</tr>
</tbody>
</table>

Table 7-5 X.25 Layer 3 Filter Commands
Triggers perform specific actions when a certain event occurs. There are four independent triggers. This section describes the commands to set the events and actions for each trigger.

8.1 Trigger Control

TR1_ON ( -- )
Turns on trigger 1.

- Triggers topic
  Arm Trig #1 function key (highlighted)

TR2_ON ( -- )
Turns on trigger 2.

- Triggers topic
  Arm Trig #2 function key (highlighted)

TR3_ON ( -- )
Turns on trigger 3.

- Triggers topic
  Arm Trig #3 function key (highlighted)

TR4_ON ( -- )
Turns on trigger 4.

- Triggers topic
  Arm Trig #4 function key (highlighted)

TR1_OFF ( -- )
Turns off trigger 1.

- Triggers topic
  Arm Trig #1 function key (not highlighted)

TR2_OFF ( -- )
Turns off trigger 2.

- Triggers topic
  Arm Trig #2 function key (not highlighted)
8-2
September 1990

TR3_0FF ( -- )
Turns off trigger 3.

Triggers topic
Arm Trig #3 function key (not highlighted)

TR4_0FF ( -- )
Turns off trigger 4.

Triggers topic
Arm Trig #4 function key (not highlighted)

8.2 Trigger Conditions

A trigger must be selected before the conditions can be specified. Trigger conditions can be specified for a specific event or any combination of events.

TR1 ( -- )
Selects trigger 1.

TR2 ( -- )
Selects trigger 2.

TR3 ( -- )
Selects trigger 3.

TR4 ( -- )
Selects trigger 4.

The selected trigger can be set to trigger on events from the network, the user, or both. The trigger can also be set to trigger on data played back from capture RAM or disk.

=NET ( -- )
Triggers on events from the NT interface direction.

Example:
Set trigger 2 to trigger on events from the network side.

TR2 =NET

=USR ( -- )
Triggers on events from the TE interface direction.

=BOTH ( -- )
Triggers on events from both interface directions.

=PLAYBACK ( -- )
Triggers on events played back from capture RAM or disk.
There are two trigger commands associated with each layer 1 event. Commands to set the event trigger are prefixed with ‘+’; commands to clear the event trigger are prefixed with ‘−’. There are no layer 1 trigger commands for the WAN interface.

**Figure 8–1 Layer 1 Event Menu (Basic Rate)**

```
Layer 1 Events:
+ ACTIVATE OFF RI2ERR OFF RECOVERY OFF
− DEACTIVATE OFF LOST FRAMING OFF UNDEFINED OFF
```

There are two trigger commands associated with each layer 1 event. Commands to set the event trigger are prefixed with ‘+’; commands to clear the event trigger are prefixed with ‘−’. There are no layer 1 trigger commands for the WAN interface.

### Table 8–1 Basic Rate Layer 1 Trigger Commands

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Set Trigger</th>
<th>Clear Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activated</td>
<td>+ACTIVATED</td>
<td>−ACTIVATED</td>
</tr>
<tr>
<td>Deactivated</td>
<td>+DEACTIVATED</td>
<td>−DEACTIVATED</td>
</tr>
<tr>
<td>Lost Framing</td>
<td>+LSTFRM</td>
<td>−LSTFRM</td>
</tr>
<tr>
<td>Recovery</td>
<td>+RECOVERY</td>
<td>−RECOVERY</td>
</tr>
<tr>
<td>INFO 2 Error</td>
<td>+RI2ERR</td>
<td>−RI2ERR</td>
</tr>
<tr>
<td>Undefined</td>
<td>+UNDEFINED</td>
<td>−UNDEFINED</td>
</tr>
</tbody>
</table>

### Table 8–2 Primary Rate Layer 1 Trigger Commands

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Set Trigger</th>
<th>Clear Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Alarm</td>
<td>+RED_ALARM</td>
<td>−RED_ALARM</td>
</tr>
<tr>
<td>Yellow Alarm</td>
<td>+YEL_ALARM</td>
<td>−YEL_ALARM</td>
</tr>
<tr>
<td>Lost Signal</td>
<td>+LST_SIGL</td>
<td>−LST_SIGL</td>
</tr>
<tr>
<td>Out of Frame</td>
<td>+OUT_OF_FRM</td>
<td>−OUT_OF_FRM</td>
</tr>
<tr>
<td>Lost Phase</td>
<td>+LST_PLOCK</td>
<td>−LST_PLOCK</td>
</tr>
<tr>
<td>Synchronized</td>
<td>+NORMAL</td>
<td>−NORMAL</td>
</tr>
</tbody>
</table>

**Example:**
Set the conditions for trigger 3 to execute when an activation or deactivation event occurs.

```
TR3
+ACTIVATED
+DEACTIVATED
```
Layer 2

Figure 8-2 Layer 2 Event Menu

There are two trigger commands associated with each layer 2 frame. Commands to set the frame trigger are prefixed with '+'; commands to clear the frame trigger are prefixed with '−'.

<table>
<thead>
<tr>
<th>Frame Type</th>
<th>Set Trigger</th>
<th>Clear Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>SABM</td>
<td>+SABM</td>
<td>−SABM</td>
</tr>
<tr>
<td>SABME</td>
<td>+SABME</td>
<td>−SABME</td>
</tr>
<tr>
<td>UA</td>
<td>+UA</td>
<td>−UA</td>
</tr>
<tr>
<td>I</td>
<td>+I</td>
<td>−I</td>
</tr>
<tr>
<td>DISC</td>
<td>+DISC</td>
<td>−DISC</td>
</tr>
<tr>
<td>REJ</td>
<td>+REJ</td>
<td>−REJ</td>
</tr>
<tr>
<td>RNR</td>
<td>+RNR</td>
<td>−RNR</td>
</tr>
<tr>
<td>RR</td>
<td>+RR</td>
<td>−RR</td>
</tr>
<tr>
<td>FRMR</td>
<td>+FRMR</td>
<td>−FRMR</td>
</tr>
<tr>
<td>DM</td>
<td>+DM</td>
<td>−DM</td>
</tr>
<tr>
<td>UI</td>
<td>+UI</td>
<td>−UI</td>
</tr>
<tr>
<td>XID</td>
<td>+XID</td>
<td>−XID</td>
</tr>
<tr>
<td>Invalid</td>
<td>+INVF</td>
<td>−INVF</td>
</tr>
</tbody>
</table>

Table 8-3 Layer 2 Trigger Commands

Additionally, the SAPI and TEI values of a frame can be set as trigger conditions.

=SAPI ( SAPI value -- )

Sets the value of the SAPI trigger.

+SAPI ( -- )

Sets the SAPI trigger.
TRIGGERS

- SAPI ( -- )
  Clears the SAPI trigger.

= TEI ( TEI value -- )
  Sets the value of the TEI trigger.

+ TEI ( -- )
  Sets the TEI trigger.

- TEI ( -- )
  Clears the TEI trigger.

+ DLCI ( -- )
  Sets both SAPI and TEI triggers. The trigger is activated when both the SAPI and TEI values of the incoming frame match the values set by =SAPI and =TEI.

- DLCI ( -- )
  Clears both SAPI and TEI triggers.

Example:
Set the conditions for trigger 4 to execute when a TEI assignment frame (SAPI 63 and TEI 127) is received.

TR4
63 =SAPI
127 =TEI
+DLCI

A specific string of characters with the received frame can be used as a trigger condition.

=STRING ( string -- )
  Specifies the string to trigger on. The string must be defined by enclosing the text characters in quotes or enclosing hex characters in quotes with a leading 'X' character. The maximum length is 64 characters.

Example:
  " ABC" =STRING
  or
  X" 0104" =STRING

+STRING ( -- )
  Sets the string trigger.

-STRING ( -- )
  Clears the string trigger.

=MASK ( string -- )
  Specifies the string mask. Bits are set to 0 to indicate "don't care" character positions. Each byte of the compare string and the received frame is ANDed with the corresponding byte of the string mask before they are compared.
Example:
Set the conditions for trigger 1 to execute when a valid frame with the C/R bit set to 1 is received.

TR1
X" 0301" =MASK
      ( only compare C/R bit and EA bits of octets 1 and 2 )
X" 0201" =STRING
      ( check for C/R = 1 and EA bits of 0 and 1 )
+STRING

---

Layer 3

Layer 3 Event Menu

<table>
<thead>
<tr>
<th>Protocol Discriminator</th>
<th>OFF</th>
<th>PD Value</th>
<th>---</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call Reference</td>
<td>OFF</td>
<td>CR Value</td>
<td>---</td>
</tr>
<tr>
<td>Message Set</td>
<td>CCITT_1988</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALERT</td>
<td>OFF</td>
<td>REL</td>
<td>OFF</td>
</tr>
<tr>
<td>CALL_PROC</td>
<td>OFF</td>
<td>REL_COM</td>
<td>OFF</td>
</tr>
<tr>
<td>CON_CON</td>
<td>OFF</td>
<td>RES</td>
<td>OFF</td>
</tr>
<tr>
<td>CONN</td>
<td>OFF</td>
<td>RES_ACK</td>
<td>OFF</td>
</tr>
<tr>
<td>CONN_ACK</td>
<td>OFF</td>
<td>RES_REJ</td>
<td>OFF</td>
</tr>
<tr>
<td>DISC</td>
<td>OFF</td>
<td>SETUP</td>
<td>OFF</td>
</tr>
<tr>
<td>INFO</td>
<td>OFF</td>
<td>SETUP_ACK</td>
<td>OFF</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>OFF</td>
<td>STATUS</td>
<td>OFF</td>
</tr>
<tr>
<td>PROG</td>
<td>OFF</td>
<td>STATUS_EN</td>
<td>OFF</td>
</tr>
</tbody>
</table>

Figure 8–3 Layer 3 Event Menu

T+MSG (message identifier -- )
Sets the message trigger for the specified message. The single input parameter is a message identifier. Valid message identifiers are listed in each message set manual.

⚠️ NOTE
The special message identifiers M#INVALID or M#UNDEF may be used to trigger on invalid or undefined messages respectively.

Example:
Set the conditions for trigger 2 to execute when a SETUP message is received.
TR2
M#SETUP   T+MSG

T–MSG (message identifier -- )
Clears the message trigger for the specified message.
The protocol discriminator and call reference fields of layer 3 messages can be used as trigger conditions.

\[ =T_{PD} \text{ (protocol discriminator value -- )} \]

Selects the protocol discriminator trigger value.

\[ T+PD \text{ (-- )} \]
Sets the protocol discriminator trigger.

\[ T-PD \text{ (-- )} \]
Clears the protocol discriminator trigger.

\[ =T_{CR} \text{ (call reference value -- )} \]
Selects the call reference trigger value.

\[ T+CR \text{ (-- )} \]
Sets the call reference trigger.

\[ T-CR \text{ (-- )} \]
Clears the call reference trigger.

---

**X.25 Layer 3**

<table>
<thead>
<tr>
<th>Packet Reception Triggers</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \rightarrow \text{Call} )</td>
</tr>
<tr>
<td>Clear</td>
</tr>
<tr>
<td>Data</td>
</tr>
</tbody>
</table>

*Figure 8-4 Packet Reception Triggers*
There are two trigger commands associated with each layer 3 frame. Commands to set the frame trigger are prefixed with ‘+’; commands to clear the frame trigger are prefixed with ‘−’.

<table>
<thead>
<tr>
<th>Packet Type</th>
<th>Set Trigger</th>
<th>Clear Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call Request</td>
<td>+CALLR</td>
<td>−CALLR</td>
</tr>
<tr>
<td>Call Connect</td>
<td>+CALLC</td>
<td>−CALLC</td>
</tr>
<tr>
<td>Clear Request</td>
<td>+CLEARR</td>
<td>−CLEARR</td>
</tr>
<tr>
<td>Clear Confirmation</td>
<td>+CLEARC</td>
<td>−CLEARC</td>
</tr>
<tr>
<td>Reset Request</td>
<td>+RESETR</td>
<td>−RESETR</td>
</tr>
<tr>
<td>Reset Confirmation</td>
<td>+RESETC</td>
<td>−RESETC</td>
</tr>
<tr>
<td>Restart Request</td>
<td>+RESTARTR</td>
<td>−RESTARTR</td>
</tr>
<tr>
<td>Restart Confirmation</td>
<td>+RESTARTC</td>
<td>−RESTARTC</td>
</tr>
<tr>
<td>Interrupt</td>
<td>+INTR</td>
<td>−INTR</td>
</tr>
<tr>
<td>Interrupt Confirmation</td>
<td>+INTC</td>
<td>−INTC</td>
</tr>
<tr>
<td>Data</td>
<td>+DATA</td>
<td>−DATA</td>
</tr>
<tr>
<td>Receive Ready</td>
<td>+RRP</td>
<td>−RRP</td>
</tr>
<tr>
<td>Receive Not Ready</td>
<td>+RNRP</td>
<td>−RNRP</td>
</tr>
<tr>
<td>Reject</td>
<td>+REJP</td>
<td>−REJP</td>
</tr>
<tr>
<td>Registration Request</td>
<td>+REGR</td>
<td>−REGR</td>
</tr>
<tr>
<td>Registration Confirmation</td>
<td>+REGC</td>
<td>−REGC</td>
</tr>
<tr>
<td>Diagnostic</td>
<td>+DIAG</td>
<td>−DIAG</td>
</tr>
<tr>
<td>Invalid</td>
<td>+INVP</td>
<td>−INVP</td>
</tr>
</tbody>
</table>

Table 8-4  X.25 Layer 3 Trigger Commands

Other Triggers

+CAPT_FULL ( -- )
Sets the capture RAM trigger. The trigger is activated when the capture RAM is full.

−CAPT_FULL ( -- )
Clears the capture RAM trigger.

+DISK_FULL ( -- )
Sets the disk trigger. The trigger is activated when the disk recording file is full.

−DISK_FULL ( -- )
Clears the disk trigger.

=TIME ( year\month\day\hour\minute -- )
Specifies the time trigger values. The trigger is activated when the specified minute is reached.

+TIME ( -- )
Sets the time trigger.
- **TIME (---)**
  Clears the time trigger.

- **NONE (---)**
  Clears all layer 1 and layer 2 triggers. Additionally, the string, time, DLCI, capture RAM, and disk triggers are each cleared.

- **ALL (---)**
  Sets all layer 1 and layer 2 triggers.

### 8.3 Trigger Actions

**Trigger Action Menu**

<table>
<thead>
<tr>
<th>Event Trigger</th>
<th>TRIGGER #1</th>
<th>Display</th>
<th>NO EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger Status</td>
<td>ARMED</td>
<td>RAM</td>
<td>TURN OFF</td>
</tr>
<tr>
<td>Beep</td>
<td>ON</td>
<td>Recording</td>
<td>TURN ON</td>
</tr>
<tr>
<td>Highlight</td>
<td>NO EFFECT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Data Display Message**
  "TRIGGER NO1 HAS FIRED"

- **User Window Message**
  ""

**Figure 8-5 Trigger Action Menu**

Trigger actions are specified by assigning commands to DOER words with the MAKE command (see the Programmer's Reference Manual for a complete description of DOER and MAKE). Each trigger has an associated DOER word for the action to execute when a trigger event occurs.

**TA1 (---)**
Performs the actions for trigger #1.

Example:
Set the action for trigger 1 to sound an audible alarm and start capture to RAM.
MAKE TA1 BEEP CAPT_ON;

**TA2 (---)**
Performs the actions for trigger #2.

**TA3 (---)**
Performs the actions for trigger #3.

**TA4 (---)**
Performs the actions for trigger #4.
The frame or event which causes the trigger action to execute can be highlighted in the Data Display Window.

**HIGHLIGHT ( -- )**  
Highlights the report in the Data Display Window.

**HIGHLIGHT=RED ( -- )**  
Selects the red background color for highlighting reports.

**HIGHLIGHT=BLUE ( -- )**  
Selects the blue background color for highlighting reports.

Example:  
Set the action for trigger 2 to highlight the frame event.  
MAKE TA2 HIGHLIGHT ;
Live or playback data is decoded by each protocol layer before it is displayed and before the test manager is executed. Each decoding layer stores information in a pool of variables for later use by either a test program or other parts of the application.

The layer 1 decoder decodes physical events and places them into variables for later access. These variables are set when a physical (layer 1) event occurs.

**L1-ID** (— address)
Contains an identifier for the layer 1 event. Possible values are:

<table>
<thead>
<tr>
<th>Basic Rate Interface:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R#ACTIVATE</td>
<td>Bus activation</td>
</tr>
<tr>
<td>R#DEACTIVATE</td>
<td>Bus deactivation</td>
</tr>
<tr>
<td>R#LSTFRM</td>
<td>Lost framing</td>
</tr>
<tr>
<td>R#RECOVERY</td>
<td>Recovery from previous error</td>
</tr>
<tr>
<td>R#RI2ERR</td>
<td>Receive INFO2 frame error</td>
</tr>
<tr>
<td>R#UNDEFINED</td>
<td>Error undefined by CCITT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Primary Rate Interface:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R#NORMAL</td>
<td>Synchronized signal</td>
</tr>
<tr>
<td>R#REDALM</td>
<td>Red alarm</td>
</tr>
<tr>
<td>R#YELALM</td>
<td>Yellow alarm</td>
</tr>
<tr>
<td>R#LSTSIGL</td>
<td>Lost signal</td>
</tr>
<tr>
<td>R#LSTPLOCK</td>
<td>Lost phase</td>
</tr>
<tr>
<td>R#OOFALM</td>
<td>Out of frame</td>
</tr>
</tbody>
</table>

**BLOCK-COUNT** (— address)
Contains the block sequence number for the layer 1 event. Every received layer 1 event is assigned a unique sequence number. BLOCK-COUNT initially contains 0 and is incremented by one each time a layer 1 event is decoded.

**PORT-ID** (— address)
Contains a 2 byte value identifying the received direction for the layer event. The lower byte indicates the network (hex value 08) or user (hex value 20) receive stream. The upper byte indicates the application processor that received the frame.

**START-TIME** (— address)
Contains the 48 bit timestamp when the layer 1 event occurred. Use with the GET_TSTAMP_MILLI or GET_TSTAMP_MICRO commands. See the Programmer’s Reference Manual.
The layer 2 decoder decodes data link layer events and places them into variables for later access. These variables are set each time a layer 2 frame is received.

**L2-POINTER** ( -- address )
Contains the pointer to the memory address (first byte) in the received frame.

**L2-LENGTH** ( -- address )
Contains the length of the received frame. This does not include the FCS (frame check sequence) bytes.

**PORT-ID** ( -- address )
Contains a 2 byte value identifying the received direction for data. The lower byte indicates the network (hex value 08) or user (hex value 20) receive stream. The upper byte indicates the application processor that received the frame.

**BLOCK-COUNT** ( -- address )
Contains the block sequence number for live data. Every received frame is assigned a unique sequence number. BLOCK-COUNT contains 0 and is incremented by one each time a new block is received.

**START-TIME** ( -- address )
Contains the 48 bit start of frame timestamp. Use with the GET_TSTAMP_MILLI or GET_TSTAMP_MICRO commands. See the Programmer's Reference Manual.

**END-TIME** ( -- address )
Contains the 48 bit end of frame timestamp. Use with the GET_TSTAMP_MILLI or GET_TSTAMP_MICRO commands.

**MADDR1** ( -- address )
Contains the first byte of the address field (octet 2) of the frame.

**MADDR2** ( -- address )
Contains the second byte of the address field (octet 3) of the frame.

**MCONTROL1** ( -- address )
Contains the first byte of the control field (octet 4) of the frame.

**MCONTROL2** ( -- address )
Contains the second byte of the control field (octet 5) of the frame. This byte is only present for frames received in modulus 128 operation.

**MSAPI** ( -- address )
Contains the SAPI value of the frame. Valid values are 0 through 63.
MTEI* ( -- address )
Contains the TEI value of the frame. Valid values are 0 through 127.

MC/R-BIT* ( -- address )
Contains the command/response bit value of the frame.

CMND/RESP* ( -- address )
Contains a 1 for command frames, and 0 for response frames.

MP/F-BIT* ( -- address )
Contains the poll/final bit value of the frame.

MNR* ( -- address )
Contains the N(R) receive sequence number of the frame.

MNS* ( -- address )
Contains the N(S) send sequence number of the frame.

MMODE-FLAG* ( -- address )
Contains a 0 if frame modulo 8, and 1 if frame modulo 128. MMODE-FLAG* is set to 0 when a SABM is decoded, and 1 when a SABME is decoded.

FRAME-ID ( -- address )
Contains the frame identifier for the last decoded frame. Possible values are:

<table>
<thead>
<tr>
<th>Frame Identifiers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R#SABM</td>
<td>Set asynchronous balanced mode frame</td>
</tr>
<tr>
<td>R#SABME</td>
<td>Set asynchronous balanced mode extended frame</td>
</tr>
<tr>
<td>R#I</td>
<td>Information frame</td>
</tr>
<tr>
<td>R#UI</td>
<td>Unnumbered information frame</td>
</tr>
<tr>
<td>R#RR</td>
<td>Receive ready frame</td>
</tr>
<tr>
<td>R#RNR</td>
<td>Receive not ready frame</td>
</tr>
<tr>
<td>R#REJ</td>
<td>Reject frame</td>
</tr>
<tr>
<td>R#DISC</td>
<td>Disconnect frame</td>
</tr>
<tr>
<td>R#UA</td>
<td>Unnumbered acknowledgement frame</td>
</tr>
<tr>
<td>R#FRMR</td>
<td>Frame reject frame</td>
</tr>
<tr>
<td>R#DM</td>
<td>Disconnected mode frame</td>
</tr>
<tr>
<td>R#XID</td>
<td>XID frame</td>
</tr>
<tr>
<td>R#INVFRM</td>
<td>Invalid frame</td>
</tr>
<tr>
<td>R#INVCRC</td>
<td>Invalid CRC in received frame</td>
</tr>
<tr>
<td>R#SHORTFRM</td>
<td>Short frame</td>
</tr>
</tbody>
</table>
STATUS_ERR? (-- flag)
Returns true if any error occurred in the last received frame. The following commands can be used to detect specific errors.

<table>
<thead>
<tr>
<th>Command</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC_ERR?</td>
<td>CRC error</td>
</tr>
<tr>
<td>ABORT_ERR?</td>
<td>Aborted frame</td>
</tr>
<tr>
<td>OVERRUN_ERR?</td>
<td>Receiver overrn</td>
</tr>
<tr>
<td>SHORT_FRM_ERR?</td>
<td>Frame is shorter than 4 bytes (including 2 CRC bytes)</td>
</tr>
<tr>
<td>CTRL_BYTE_ERR?</td>
<td>Invalid control byte(s)</td>
</tr>
<tr>
<td>ADDR_BYTE_ERR?</td>
<td>Invalid address byte(s)</td>
</tr>
<tr>
<td>LENGTH_ERR?</td>
<td>Invalid frame length</td>
</tr>
<tr>
<td>XID_INFO_ERR?</td>
<td>Invalid information field in XID frame</td>
</tr>
<tr>
<td>FRMR_INFO_ERR?</td>
<td>Invalid information field in FRMR frame</td>
</tr>
</tbody>
</table>

10.1 FRMR Frames

The following variables can be set when an FRMR frame is decoded.

FRMR-CNTL1* (-- address)
Contains the first byte of the control field (octet 5) of the FRMR information field.

FRMR-CNTL2* (-- address)
Contains the second byte of the control field (octet 6) of the FRMR information field.

FRMR-VR* (-- address)
Contains the value of the V(R) state variable field of the FRMR information field.

FRMR-VS* (-- address)
Contains the value of the V(S) state variable field of the FRMR information field.

FRMR-C/R-BIT* (-- address)
Contains the value of the C/R bit of the FRMR information field.

FRMR-W-BIT* (-- address)
Contains the value of the W bit of the FRMR information field.

FRMR-X-BIT* (-- address)
Contains the value of the X bit of the FRMR information field.

FRMR-Y-BIT* (-- address)
Contains the value of the Y bit of the FRMR information field.

FRMR-Z-BIT* (-- address)
Contains the value of the Z bit of the FRMR information field.
10.2 UI Frames

The following variables are set when a UI frame is decoded.

**UI-TYPE** ( -- address )
Contains the UI frame identifier. Possible values are:

- TEI-ASSIGN#
  - SAPI = 63 and TEI = 127
- MDL-UNIT-DATA#
  - SAPI = 63 and TEI ≠ 127
- DL-UNIT-DATA#
  - SAPI ≠ 63

**MANAGEMENT-ID** ( -- address )
Contains the management entity identifier of the TEI assignment UI frame.

**RI** ( -- address )
Contains the reference number of the TEI assignment UI frame.

**MESSAGE-TYPE** ( -- address )
Contains the message type of the TEI assignment UI frame. Possible values are:

- ID-REQ#
  - Identity request
- ID-ASSGN#
  - Identity assigned
- ID-DENIED#
  - Identity denied
- ID-CHK-REQ#
  - Identity check request
- ID-CHK-RESP#
  - Identity check response
- ID-REMOVE#
  - Identity remove
- ID-VERIFY#
  - Identity verify

**AI** ( -- address )
Contains the action indicator of the TEI assignment UI frame.

10.3 XID Frames

The following variables are set when an XID frame is decoded.

**XID-TX-N201** ( -- address )
Contains the value of the transmit N201 field of the XID frame.

**XID-RX-N201** ( -- address )
Contains the value of the receive N201 field of the XID frame.

**XID-K** ( -- address )
Contains the value of the transmit window size field of the XID frame.

**XID-T200** ( -- address )
Contains the value of the retransmission timer field of the XID frame.
The layer 3 message decoder disassembles signalling messages and stores the decoded protocol information into variables. This protocol information includes the protocol discriminator, call reference, message type, and all parameters contained in each information element within the received message.

Messages can contain information elements, some of which are mandatory and others which are optional or not allowed. The sequence of information elements for a particular codeset is pre-defined but can be interspersed with information elements from other codesets. Furthermore, multiple information elements of the same type can occur in a message and can have a variable structure depending on inclusion or exclusion of particular octets. To simplify the method by which this information is obtained, the layer 3 decoder provides functions to determine:

- if all mandatory information elements are present within a message;
- if any optional information elements are present within a message;
- if any unexpected information elements are present within a message;
- if all information elements occur in the correct order within the message;
- how often a particular information element occurs within a message;
- if it is possible to decode the \( n \)th occurrence of an information element;
- if a particular octet was present in the information element; and
- the type of error that occurred while decoding the information element.

The decoded value for each field of every information element can be accessed by a unique identifier. Additionally, the layer 3 decoder uses pre-defined constants to identify particular information element field values. These field identifiers and constants are unique for each message set.

### 11.1 Variables

The following variables contain layer 3 information for the received message.

**L3-POINTER** ( -- address )

Contains the memory address of the first byte in the received message.

**L3-LENGTH** ( -- address )

Contains the length of the received message.
$MSG-ERROR ( -- address )
Contains an error code identifying the error that occurred when decoding the last received message. It can be one of the following defined constants:

- **NO-ERROR#**  No error was detected
- **SHORT-MSG#** Received message was too short. Each message header must have a protocol discriminator, call reference, and message type field
- **INV-DISCR#** Unknown protocol discriminator
- **INV-CALLREF#** Call reference has invalid format
- **SHIFT-ERR#** Shift IE after non-locking shift detected
- **RANGE-ERR#** Information element exceeds message range
- **NO-MSG#** No message received. The value of L3-LENGTH is zero

\*\* NOTE \*\*
Depending on the value of $MSG-ERROR, not all decoder variables contain valid information (i.e. when the protocol discriminator is unknown, there is no valid entry in $MSG-TYPE).

$MSG-DISCR ( -- address )
Contains the protocol discriminator value of the received message.

$MSG-CRLEN ( -- address )
Contains the call reference length of the received message.

$MSG-CRFLAG ( -- address )
Contains the call reference flag of the received message. Possible values are #ORIG (0) and #DEST (1).

$MSG-CRVALUE ( -- address )
Contains the call reference value of the received message.

$MSG-ID ( -- address )
Contains the message identifier of the received message. A complete list of possible message identifiers is given in the appropriate message set manual.

\*\* NOTE \*\*
This value is different from the message type. The message identifier is a constant that uniquely identifies the message; the message type is the actual value that appears in the message header.

$MSG-TYPE ( -- address )
Contains the message type of the received message. This is the actual bit value of the received message.
11.2 Commands

The following commands determine the type and content of received layer 3 messages.

?L3_MSG (message identifier -- flag)
Returns true if the specified message is received. The message is identified by a symbolic name or by one of three special identifiers:

- M#ANY: Any message
- M#INVALID: Invalid message
- M#UNDEF: Undefined message

Example:
Check for the reception of a PROGRESS message (CCITT message set).

\[
\ldots
\]
M#PROGRESS ?L3_MSG
ACTION{
   < perform action required if PROGRESS message received >
\}

?L3_VALID_MSG (message identifier -- flag)
Returns true if the specified message was the last one received and was coded correctly. A message is accepted as correctly coded if the order of information elements is correct, no mandatory IE's are missing, no unexpected IE's are present, and the message type is valid for the received direction.

\[\triangledown\] NOTE
The special message identifiers M#ANY, M#INVALID, and M#UNDEF may only be used with the ?L3_MSG and ?L3_VALID_MSG commands. M#ANY is used as a 'wildcard' value for any message with a valid message header.

?L3_MATCH (string -- flag)
Returns true if the user-defined key string is found in the layer 3 message.

This is an anchored match, i.e. a byte-for-byte match starting at the first byte of the received message.

\[\triangledown\] NOTE
To accommodate "don't care" character positions, the question mark character for ASCII or hex 3F character can be used. The received message can be longer than the specified key string.
Example:
Format for string matching within messages.

```
" key string" ?L3_MATCH
ACTION{
   < action to be taken if string matches >
}

?L3_SEARCH ( string -- flag )
Returns true if the user-defined key string is found in the layer 3 message.

This is an unanchored match, i.e. searches for an exact match anywhere in the received
message regardless of the position.

?L3_IE_ORDER ( -- flag )
Returns true if all information elements occur in correct order within the received message. If
multiple IE's of the one type occur within the message, they must be grouped together.

?L3_MAND_IE ( message identifier -- flag )
Returns true if all mandatory information elements were found for the specified message.

?L3_OPT_IE ( message identifier -- flag )
Returns true if at least one optional information element was found for the specified message type.

?L3_UNEXP_IE ( message identifier -- flag )
Returns true if at least one unexpected information element was found for the specified
message type. An information element is considered to be unexpected when it is neither
mandatory nor optional, or when it is mandatory or optional but illegally occurs multiple times
within a message.

Example:
Check IE occurrences within a received message.

M#SETUP ?L3_UNEXP_IE ( Check for unexpected IEs )
?L3_IE_ORDER 0= OR ( Or wrong sequence of IEs )
IF
   < handle Setup message containing unexpected or badly placed IEs >
ELSE
   M#SETUP ?L3_MAND_IE 0= ( Check for missing mandatory IEs )
   IF
      < handle Setup message with missing mandatory IEs >
   ELSE
      < handle Setup message with all required IEs >
   M#SETUP ?L3_OPT_IE ( Check for optional IEs )
   IF
      < handle optional IEs of Setup message >
ENDIF
ENDIF
ENDIF
NOTE
Shift IE's are not considered to be unexpected.

?L3_MAND_ERROR (message identifier -- flag)
Returns true if there was an error in one of the mandatory IE's for the specified message.

?L3_OPT_ERROR (message identifier -- flag)
Returns true if there was an error in one of the optional IE's for the specified message.

?L3_MSG_DIR (message identifier -- flag)
Returns true if the specified message was received in the correct direction (user -> network or network -> user).

?L3_IECOUNT (IE identifier -- count)
Returns the number of times the specified information element was found in the received message.

Example:
Check the presence of the Bearer Capability IE (CCITT message set).

I#BEARER_CAP ?L3_IECOUNT 0=
IF
   < Bearer Capability IE was not present in message >
ELSE
   < Process Bearer Capability IE >
ENDIF

?L3_IE (IE identifier -- flag)
Returns true if the specified information element occurs at least 'n' times in the message. If present, the n-th occurrence of the IE is decoded for later access of parameter values.

Example:
Decode the third Cause IE.

I#CAUSE 3 ?L3_IE
IF
   < perform action required if 3rd CAUSE information element found >
ELSE
   < perform action required if 3rd CAUSE information element not found >
ENDIF

NOTE
?L3_IE must be used before parameter values of an information element can be accessed.
?L3_IE_ERROR ( IE identifier -- code )
Returns an error code indicating which error occurred while decoding the specified
information element. Possible values are:

- NO-IE-ERROR#  No error occurred
- IE-EMPTY#     IE length is zero
- IE-TOO-SHORT# IE content is too short
- IE-TOO-LONG#  IE content is too long
- UNEXP-EXTEND# Expected extension bit was not set
- EXP-EXTEND#   Unexpected extension bit
- PARAM-WRONG-LEN# String parameter exceeds maximum length

?L3_OCTET ( IE identifier \ octet identifier -- n )
Returns the number of times the specified octet occurred within the specified information
element. For octets that do not repeat in an information element, this value will be 0 or 1.
Valid octet identifiers for the 1988 CCITT Blue Book message set are listed below:

- OCTET_3
- OCTET_3.3
- OCTET_5A
- OCTET_6A
- OCTET_3A
- OCTET_4
- OCTET_5B
- OCTET_7
- OCTET_3B
- OCTET_4A
- OCTET_6
- OCTET_3.1
- OCTET_4B
- OCTET_8
- OCTET_3.2
- OCTET_5
- OCTET_5D
- OCTET_6

Example:
Check if octet 4b is present in BEARER_CAP.

I#BEARER_CAP OCTET_4B ?L3_OCTET
IF
  < process Symmetry and Transfer Rate field values >
ENDIF

NOTE
?L3_OCTET must be used to verify the presence of the specified octet(s) before parameter
values can be accessed.
Example:
Access the parameters in the Cause IE (CCITT message set).

```
I#CAUSE ?L3_IECOUNT                      ( Check if cause IE is present )
IF
  I#CAUSE 1 ?L3_IE                       ( If yes, decode first occurrence )
  IF
    *DEC ->C_LOCATION @                 ( Process location code )
    ...
    I#CAUSE OCTET_3A ?L3_OCTET          ( Check if octet 3a was included )
    IF
      *DEC ->C_RECOMMENDATION @         ( Process recommendation parameter )
      ...
    ENDIF
    *DEC ->C_CAUSE_VALUE @              ( Process cause value )
    ...
  ELSE
    < handle decoding error >
  ENDIF
ELSE
  < handle absence of Cause information element >
ENDIF

*DEC ( -- address )
Returns the base address of the decoder parameter structure. *DEC is used in conjunction
with the parameter selector commands to access the decoded parameter values.

The parameter value may be compared to one of several pre-defined constants defined for
that parameter field. Some parameter fields represent string values. These strings can be
printed out using the standard output commands or copied to other buffers. See the Strings

Selector commands and pre-defined constants are unique for each message set. See the corresponding ISDN Message Sets Reference Manual or Section 14 of this manual for more
information.
Example 1:
Determine the B-Channel allocated to a voice call, as specified within the Channel ID Information element, for a CCITT message. This example uses identifiers from the CCITT message set.

```plaintext
*DEC ->CID_INFO_CHAN_SEL @ ( Get constant value )
```

```plaintext
DOCASE
    CASE #NO_CHANNEL
        T. " No channel is available" TCR
    
    CASE #B1_CHANNEL
        T. " Channel number 1 selected" TCR
    
    CASE #B2_CHANNEL
        T. " Channel number 2 selected" TCR
    
    CASE DUP
        T. " Channel not explicitly identified" TCR

ENDCASE
```

Example 2:
Print the value of the calling number.
T. " The number is:

```plaintext
T. ClGN_NUMBER COUNT T.TYPE TCR
```

**NOTE**
?

*L3_IE must be used to decode the information element before parameter values can be accessed. Also, it must be used to verify the presence of the specified octet(s) before parameter values can be accessed.*

Some octets in certain IE's can be repeated. The parameter values for these octets are stored in an array and must be accessed with the [] command.

Example:
Print the feature status values from the Feature Indication IE in the AT&T 5E6 message set.

```plaintext
OCTET_5 ?L3_OCTET 2 =

IF
    *DEC ->FI_STATUS 1 [] @ T.
    *DEC ->FI_STATUS 2 [] @ T.
ENDIF
```

**NOTE**
?

*The array index for repeated parameter values starts at 1. The following expressions are identical for accessing the first value in the array:

*DEC ->FL_STATUS or *DEC ->FL_STATUS 1 [] @.*
Monitored layer 3 information in SAPI 16 I frames is decoded as X.25 layer 3 information. This decoding is applicable only to Basic Rate applications.

The layer 3 decode operation saves information concerning a packet's logical group and channel number, Q and D bits, packet identifier, and fields pertinent to specific packets. The pointer to and the length of the data field is determined for data packets.

M-GFI ( -- address )
Contains the GFI (general format identifier) value of the received packet. Valid values are 1 for packet modulo 8, and 2 for a packet modulo 128. These are bits 5 through 8 of the first packet octet.

M-Q ( -- address )
Contains the Q (qualifier) bit of a received data packet (0 or 1). This is bit 8 of the first data packet octet.

M-D ( -- address )
Contains the D (delivery confirmation) bit of a received data packet, call request/incoming call, or call accepted/confirmed packet (0 or 1). This is bit 7 of the first packet octet.

M-LCG ( -- address )
Contains the logical group number of the received packet. Valid values are 0 through 15. These are bits 1 through 4 of the first packet octet.

M-LCB ( -- address )
Contains the logical channel number of the received packet. Valid values are 0 through 255. This is the second packet octet.

M-LCN ( -- address )
Contains the combined logical group identifier and logical channel number of the received packet. Valid values are 0 through 4095. These are bits 1 through 4 of the first packet octet and all bits of the second packet octet.

M-REC-PKT-ID ( -- address )
Contains the packet type identifier. This is the third packet octet. See Table B-2 for possible values.

M-RCAUSE ( -- address )
Contains the cause byte. Valid values are 0 through 255. This is octet 4 of the following packets:
- Clear request/indication
- Reset request/indication
- Restart request/indication
M-RIAG ( -- address )
Contains the diagnostic byte of the received packet. Valid values are 0 through 255. This is octet 5 of the following packets:
- Clear request/indication
- Reset request/indication
- Restart request/indication

M-MORE ( -- address )
Contains the more bit of a received data packet (0 or 1). Bit 5 of the third octet of data packet for modulo 8, and bit 1 of the fourth octet of data packet for modulo 128.

M-PS ( -- address )
Contains the P(S) (send sequence) count of the received packet. Valid values are 0 through 7 for packet modulo 8, and 0 through 127 for packet modulo 128. The P(S) counts are bits 2 through 4 of the third octet of data packet for modulo 8, and bits 2 through 8 of the third octet of data packet for modulo 128.

M-PR ( -- address )
Contains the P(R) (received sequence) count of the received packet. Valid values are 0 through 7 for packet modulo 8, and 0 through 127 for packet modulo 128. The P(R) counts are bits 6 through 8 of the third octet of data, RR, RNR, and REJ packets for modulo 8, and bits 2 through 8 of the fourth octet of data, RR, RNR, and REJ packets for modulo 128.

DATA-POINTER ( -- address )
Contains the pointer to the first character of the data field in any layer 4 or user data present in a data packet.

DATA-LENGTH ( -- address )
Contains the length of the data field in a received data packet.

M-RCALLED ( -- address )
Contains a 16 byte string identifying the called address field of the received packet. See M-RCALLING for an example; string contents and handling are similar.
**M–RCALLING ( address )**
Contains a 16 byte string identifying the calling address field of the received packet.

Example:
Check whether the CALLING number of the last received call request packet matches a pre-defined number: 43042001. The ?MATCH command is used to determine if the calling address in a received call request packet matches the defined address.

```
TCLR

#IFDEF MATCH-CALL
    0 VARIABLE MATCH-CALL 12 ALLOT
#ENDIF  ( MATCH-CALL will contain the desired calling address )

0 STATE_INIT{
    "43042001"
    COUNT  ( Obtain # of digits in calling address )
    15 MIN  ( Set maximum to 15 digits )
    DUP MATCH-CALL C!  ( Write # of digits to first byte of matching string )
    MATCH-CALL 1 + SWAP CMOVE  ( Write calling address in matching string )
}STATE_INIT

0 STATE{
    R*CALLREQ 1 ?RX
    ACTION{
        M–RCALLING  ( Get this calling address )
        COUNT  ( Get # of digits in this calling address )
        MATCH-CALL  ( Get desired matching address )
        ?MATCH  ( Compare addresses )
        IF
            BEEP  ( Notify user )
            T."Address Match has occurred."  TCR
        ENDIF
    }ACTION
}STATE
```
M-RFAC ( -- address )
Contains a 128 byte string identifying the facility field of the last received call request/incoming call, call accept/connect, clear request/indication, or clear confirmation packet.

The first byte of this string contains the length of the facility field. Valid values are 0 through 63 for 1980, and 0 through 109 for 1984.

Example:
The tester receives a call request packet with a facility field that contains eight characters, i.e. the hex characters 02AA4208084303.

Obtain the length of the facility field (8 in this case).
M-RFAC C@

Obtain the first octet of the first facility (hex 02 in this case). This indicates:
• the first facility is a Class A facility, i.e. it is followed by a single octet parameter field;
  and
• the first facility is throughput class negotiation.
M-RFAC 1 + C@

Obtain the throughput class octet (hex AA in this case). This indicates that the throughput class for the called DTE and calling DTE is 9600 bits.
M-RFAC 2 + C@

Obtain the first octet of the second facility (hex 42 in this case). This indicates:
• the second facility is a Class B facility, i.e. it is followed by a 2 octet parameter field;
  and
• the second facility is packet size.
M-RFAC 3 + C@

Obtain the packet size for the called DTE (hex 08 in this case). This indicates a packet size of 256.
M-RFAC 4 + C@

Obtain the packet size for the calling DTE (hex 08 in this case). This indicates a packet size of 256.
M-RFAC 5 + C@

Obtain the first octet of the third facility (hex 43 in this case). This indicates:
• the third facility is a Class B facility, i.e. it is followed by a 2 octet parameter field; and
  the third facility is window size.
M-RFAC 6 + C@

Obtain the window size of the called DTE (hex 03 in this case). This indicates a window size of 3.
M-RFAC 7 + C@

Obtain the window size of the calling DTE (hex 03 in this case). This indicates a window size of 3.
M-RFAC 8 + C@
NOTE
Refer to CCITT X.25 (1984) Recommendation, Section 7, formats for facility fields and registration fields for further information on decoding facilities.

M-RCUD ( -- address )
Contains a 256 byte string identifying the call user data of the received packet. Valid values are 0 through 128 if fast select facility is selected, and 0 through 16 if not selected.

Example:
The tester receives a call request packet with a call user field that contains eleven characters, i.e. the hex characters C000000003010025800064. M-RCUD can be used to obtain the following information.

Obtain the length of the call user data field (11 in this case).
M-RCUD C@

If the call user data field is present, its use and format are determined by bits 7 and 8 of the first octet.
M-RCUD 1 + C@

NOTE
Refer to CCITT 1984 Recommendation X.244 for further information on call user data.
A variety of message sets can be used for layer 3 message decoding and building (transmission). Message sets contain identifiers for defined message types and information elements. Additionally, selector words for each parameter field, and pre-defined constants for each parameter value are contained in the message set.

Message sets are stored in binary files unique for each application and are automatically suffixed with filename extensions. Tables 13-1 and 13-2 list the extensions for monitor and emulation, respectively, depending on the application processor and machine configuration.

Customized message sets can be created with MDL (Message Description Language) and saved to binary files for future use. Refer to the ISDN MDL Programmer’s Manual for more information.

<table>
<thead>
<tr>
<th>AP #1</th>
<th>Mm1</th>
<th>Mm1</th>
<th>Mm1</th>
<th>Mm3</th>
<th>Mm1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP #2</td>
<td>Mm2</td>
<td></td>
<td></td>
<td>Mm4</td>
<td></td>
</tr>
<tr>
<td>AP #3</td>
<td></td>
<td>Mm7</td>
<td></td>
<td>Mm7</td>
<td>Mm3</td>
</tr>
<tr>
<td>AP #4</td>
<td></td>
<td></td>
<td></td>
<td>Mm3</td>
<td></td>
</tr>
<tr>
<td>AP #5</td>
<td></td>
<td></td>
<td></td>
<td>Mm4</td>
<td></td>
</tr>
<tr>
<td>AP #6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mm5</td>
</tr>
</tbody>
</table>

Table 13–1 Message Set Filename Extensions – Monitor Applications

<table>
<thead>
<tr>
<th>AP #1</th>
<th>Me1</th>
<th>Me1</th>
<th>Me1</th>
<th>Me3</th>
<th>Me1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP #2</td>
<td>Me2</td>
<td></td>
<td></td>
<td>Me4</td>
<td></td>
</tr>
<tr>
<td>AP #3</td>
<td></td>
<td>Me7</td>
<td></td>
<td>Me7</td>
<td>Me3</td>
</tr>
<tr>
<td>AP #4</td>
<td></td>
<td></td>
<td></td>
<td>Me3</td>
<td></td>
</tr>
<tr>
<td>AP #5</td>
<td></td>
<td></td>
<td></td>
<td>Me4</td>
<td></td>
</tr>
<tr>
<td>AP #6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Me5</td>
</tr>
</tbody>
</table>

Table 13–2 Message Set Filename Extensions – Emulation Applications

Example:
The filename for the CCITT_1988 message set for the ISDN Monitor application for AP#3 of a BRA/WAN tester is CCITT_1988.Mm7.
Use LOAD_MESSAGE_SET at the beginning of a layer 3 simulation test script to ensure the correct message set is loaded before the remainder of the test script is compiled.

LOAD_MESSAGE_SET (string -- )
Loads the specified message set from disk. If not found, the error message 'could not be located on any drive' will be displayed.

Example:
TCLR
"ATT_5E6" LOAD_MESSAGE_SET

⚠️ NOTE
Do not specify the filename extension in the string.

MCLR ( -- )
Clears the current message set from memory.
The CCITT message set is implemented in accordance with: CCITT Recommendations Q.931 and Q.932 as described in the Volume VI – Fascicle VI.11 Blue Book (Nov. 1988).

The CCITT (1988) Message Set is the default message set loaded when the application starts, and contains unique identifiers which can be used in ISDN test scripts to reference received and transmitted messages. These identifiers are divided into three categories:

- Message Type Identifiers
- Information Element Identifiers
- Information Element Structures (including parameter field selectors and associated field values constants)

The message set name (eg. CCITT_1988), description, and release number (as indicated above) uniquely identify the message set variation. The message set name is used with the LOAD_MESSAGE_SET command or the Load Message Set function key under the MessageSet topic. This name is also displayed on various menus and is used to identify the message set variation when layer 3 complete report format is selected.

The next three sections provide some examples illustrating the use of each of these types of identifiers. The remaining sections contain:

- message identifiers;
- IE identifiers; and
- IE structures.

### 14.1 Using Message Identifiers

Message identifiers uniquely identify a message type in both received and transmitted messages, and are expressed in the following form:

\[ M\#xxxx \text{ (eg. } M\#SETUP) \]

In addition, the following default identifiers (specific received messages only) are also included with each message set:

- \( M\#ANY \) (any valid message)
- \( M\#INVALID \) (an invalid message)
- \( M\#UNDEF \) (an unknown/undefined message type)
Example 1:
After receiving a Setup message, perform an action (e.g. send a Setup Acknowledge response, increment a counter, etc.).

M#SETUP ?L3_MSG
ACTION{
  ( code specifying action taken if Setup message received )
}ACTION

Example 2:
Send an Alert message in an I frame complete with desired information elements.

M#ALERT MESSAGE>
  I#DISPLAY
  I#SIGNAL
<SEND

Message identifiers can also be used for filter/trigger management from within a script.

Example 3:
Set the display/report filter to only pass Setup and Connect messages.

R_FILTER
  F3=NONE
  M#SETUP F+MSG
  M#CONN F+MSG

14.2 Using IE Identifiers

IE identifiers uniquely identify an information element in both received and transmitted messages, and are expressed in the following form:

  l#xxxx (eg. l#CAUSE)

Example 1:
Determine if the Cause IE appears in the last received message at least once.

I#CAUSE 1 ?L3_IE
IF
  ( code specifying action taken if the first Cause IE is found )
ELSE
  ( code specifying action taken if the first Cause IE is not found; ie: none present )
ENDIF
Example 2:
Prepare a Cause IE for later inclusion and transmission within a message.

```
I#CAUSE ELEMENT>
   ALL_EXCLUDED
   OCTET_3 INCLUDED
   OCTET_4 INCLUDED
   OCTET_5 INCLUDED
<ELEMENT
```

Also in this group are octet identifiers which uniquely identify an octet number that can be used for any IE that contains that octet number. Octet identifiers are used in both received and transmitted messages and are expressed in the following form:

```
OCTET_xx (eg. OCTET_3.1)
```

Example 1:
Determine if Octet 3A is present in the Cause IE of the latest message received.

```
I#CAUSE OCTET_3A ?L3_OCCET
IF
   ( code specifying action taken if the octet is present;
     ie: process the specified Recommendation )
ENDIF
```

### 14.3 Using IE Structures

Information element structures consist of the information element parameter field selectors and the associated field value identifiers.

The parameter field selectors are expressed in the following form:

```
->xxx_yyyy (eg. ->BC_CODING_STANDARD)
```

where:  
xxx = the information element associated with that parameter field 
(eg: Bearer Capability) 
yyyy = the parameter field (either a string or a bit field)

The field value identifiers are expressed in the following form:

```
#xxxxx (eg. #INTERNATIONAL = 0b00000001)
```

All parameter field selectors are used with the *DEC and *COD structure indicators. *DEC provides the base address of the decoder parameter structure. When used with a field selector, decoded parameter values can be accessed. *COD complements *DEC and provides the base address of the coder parameter structure for the current connection. The contents of specific parameter fields can then be changed prior to transmission.
Example 1:
Depending on the contents of the received Bearer Capability Coding Standard parameter field (Octet 3, 2 bits), perform one of two different actions.

\*DEC \rightarrow BC\_CODING\_STANDARD @ \quad \text{(Obtain the received value)}
\#CCITT = \quad \text{(Compare with identifier)}
IF
\quad T.\_ Coding Standard is CCITT \quad \text{TCR}
ELSE
\quad T.\_ Coding Standard is not CCITT \quad \text{TCR}
ENDIF

\begin{itemize}
\item \textbf{NOTE}
\end{itemize}

\begin{itemize}
\item The preceding example uses a bit field and @ (fetch); ! (store) and T. (print value) can also be used. If the parameter is a string (a sequence of one or more characters), !STRING or T\_TYPE can be used.
\end{itemize}

Example 2:
Set the appropriate values of the two parameter fields of Octet 4 of the Bearer Capability IE prior to transmission.

\#CIRCUIT\_MODE \*COD \rightarrow BC\_TRANSFER\_MODE !
\#384KBIT/S \*COD \rightarrow BC\_TRANSFER\_RATE !

### 14.4 Message Identifiers

**Q.931 Protocol Discriminator (value = 0X08)**

- M\#ALERT \hspace{2cm} Alerting
- M\#CALL\_PROC \hspace{2cm} Call Proceeding
- M\#CONN \hspace{2cm} Connect
- M\#CONN\_ACK \hspace{2cm} Connect Acknowledge
- M\#CON\_CON \hspace{2cm} Congestion Control
- M\#DISC \hspace{2cm} Disconnect
- M\#FAC \hspace{2cm} Facility
- M\#HOLD \hspace{2cm} Hold
- M\#HOLD\_ACK \hspace{2cm} Hold Acknowledge
- M\#HOLD\_REJ \hspace{2cm} Hold Reject
- M\#INFO \hspace{2cm} Information
- M\#NOTIFY \hspace{2cm} Notify
- M\#PROG \hspace{2cm} Progress
- M\#REG \hspace{2cm} Register
- M\#REL \hspace{2cm} Release
- M\#REL\_COM \hspace{2cm} Release Complete
- M\#RES \hspace{2cm} Resume
- M\#REST \hspace{2cm} Restart
- M\#REST\_ACK \hspace{2cm} Restart Acknowledge
### 14.5 IE Identifiers

#### Codeset 0

- `#BEARER_CAP` Bearer Capability
- `#CALLED_NUM` Called Party Number
- `#CALLED_SAD` Called Party Subaddress
- `#CALLING_NUM` Calling Party Number
- `#CALLING_SAD` Calling Party Subaddress
- `#CALL_ID` Call Identity
- `#CALL_STATE` Call State
- `#CAUSE` Cause
- `#CHANNEL_ID` Channel Identification
- `#CONG_LEVEL` Congestion Level
- `#DATE/TIME` Date/time
- `#DISPLAY` Display
- `#END-END_DELAY` End-to-end Transit Delay
- `#ENDPOINT_ID` Endpoint Identifier
- `#FACILITY` Facility
- `#FEAT_ACT` Feature Activation
- `#FEAT_IND` Feature Indication
- `#HIGH_LAY_COMP` High Layer Compatibility
- `#INFO_RATE` Information Rate
- `#INFO_REQ` Information Request
- `#KEYPAD` Keypad
- `#LOW_LAY_COMP` Low Layer Compatibility
- `#MORE_DATA` More Data
- `#NETWORK_FACIL` Network Facilities
- `#NOTIFIC_IND` Notification Indicator
- `#PKT_PARAMS` Packet Layer Binary Parameters
- `#PKT_SIZE` Packet Size
- `#PKT_WINDOW` Packet Layer Window Size
- `#PROGRESS_IND` Progress Indicator
- `#REDIRING_NUM` Redirecting Number
I#REPEAT_IND  Repeat Indicator
I#RESTART_IND  Restart Indicator
I#SEGMENT  Segmented Message
I#SEND_COMP  Sending Complete
I#SHIFT  Shift
I#SIGNAL  Signal
I#SPID  Service Profile Id.
I#SWITCHHOOK  Switchhook
I#TRANS_DEL_SEL  Transit Delay Selection
I#TRANS_NW_SEL  Transit Network Select
I#UU_INFO  User–user Information

Codeset 5
I#SHIFT  Shift

Codeset 6
I#SHIFT  Shift

Codeset 7
I#SHIFT  Shift

14.6 IE Structures

Bearer Capability IE (I#BEARER_CAP)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4, OCTET_4A, OCTET_4B, OCTET_5, OCTET_5A, OCTET_5B, OCTET_5C,
OCTET_5D, OCTET_6, OCTET_7

->BC_CODING_STANDARD  Coding standard, Octet 3
        #CCITT  CCITT
        #INTERNATIONAL  other international standards
        #NATIONAL  national standard
        #NETWORK_SPECIFIC  standard defined for the network
-->

BC_TRANSFER_CAP

#SPEECH
#UNRESTRICTED
#RESTRICTED
#3.1KHZ_AUDIO
#7KHZ_AUDIO
#VIDEO

-->

BC_TRANSFER_MODE

#CIRCUIT_MODE
#PACKET_MODE

-->

BC_TRANSFER_RATE

#PACKET
#64KBIT/S
#2X64KBIT/S
#384KBIT/S
#1536KBIT/S
#1920KBIT/S

-->

BC_STRUCTURE

#DEFAULT
#8KHZ_INTEGRITY
#SDU_INTEGRITY
#UNSTRUCTURED

-->

BC_CONFIGURATION

#POINT_TO_POINT

-->

BC_ESTABLISHMENT

#DEMAND

-->

BC_SYMMETRY

#BIDIRECT_SYMMETRIC

-->

BC_TRANSFER_RATE_4B

#PACKET
#64KBIT/S
#2X64KBIT/S
#384KBIT/S
#1536KBIT/S
#1920KBIT/S

-->

BC_LAYER1_ID

(numeric value)

-->

BC_L1_PROTOCOL

#RATE_ADAPTATION
#G.711_ULAW
#G.711_ALAW
#G.721_ADPCM
#G.7XX_AUDIO
#G.7XX_VIDEO
#NON-CCITT
#V.120
#X.31_HDLC

-->

BC_SYNC/ASYNC

#SYNCHRONOUS
#ASYNCHRONOUS

Info. trans. cap., Octet 3

speech
unrestricted digital information
restricted digital information
3.1 kHz audio
7 kHz audio
video

Transfer mode, Octet 4

circuit mode
packet mode

Info. transfer rate, Octet 4

packet mode call
64 kbit/s
2 x 64 kbit/s
384 kbit/s
1536 kbit/s
1920 kbit/s

Structure, Octet 4a

default
8 kHz integrity
service data unit integrity
unstructured

Configuration, Octet 4a

point-to-point

demand

Symmetry, Octet 4b

bidirectional symmetric

Info. transfer rate, Octet 4b

packet mode call
64 kbit/s
2 x 64 kbit/s
384 kbit/s
1536 kbit/s
1920 kbit/s

Layer identifier, Octet 5

valid value: 1

Layer 1 protocol, Octet 5

CCITT rate adaption V.110/X.30
Rec. G.711 u-law
Rec. G.711 A-law
Rec. G.721 32 kbits/s ADPCM
Rec. G.722 and G.724 7kHz audio
Rec. G.7XX 384 kbit/s video
non-CCITT rate adaption
CCITT rate adaption V.120
CCITT rate adaption X.31 HDLC

Sync/Async, Octet 5a

synchronous
asynchronous
Negotiation, Octet 5a

in-band negotiation not possible
in-band negotiation possible

User rate, Octet 5a

indicated by E-bits Rec. I.460

0.6 kbit/s Rec V.6 and X.1
1.2 kbit/s Rec V.6
2.4 kbit/s Rec V.6 and X.1
3.6 kbit/s Rec V.6
4.8 kbit/s Rec V.6 and X.1
7.2 kbit/s Rec V.6
8 kbit/s Rec I.460
9.6 kbit/s Rec V.6 and X.1
14.4 kbit/s Rec V.6
16 kbit/s Rec I.460
19.2 kbit/s Rec V.6
32 kbit/s Rec I.460
48 kbit/s Rec V.6 and X.1
56 kbit/s Rec V.6

Intermediate rate, Octet 5b

not used

8 kbit/s
16 kbit/s
32 kbit/s

NIC on Tx, Octet 5b

data not required
data required

NIC on Rx, Octet 5b
cannot accept data
can accept data

Flow Control on Tx, Octet 5b

not required

Flow Control on Rx, Octet 5b
cannot accept
can accept

Rate adaption, Octet 5b

header not included
header included
-> BC_MULTI_FRAME
 #NOT_SUPPORTED
 #SUPPORTED
 -> BC_OPER_MODE
 #BIT_TRANSPARENT
 #PROT_SENSITIVE
 -> BC_LLI_NEG
 #DEFAULT
 #FULL_NEGOTIATION
 -> BC_ASSIG
 #ASSIGNEE
 #ASSIGNOR
 -> BC_BAND_NEG
 #WITH_INFO
 #WITH_LL0
 -> BC_STOP_BITS
 #NOT_USED
 #1_STOP_BIT
 #1.5_STOP_BITS
 #2_STOP_BITS
 -> BC_DATA_BITS
 #NOT_USED
 #5_DATA_BITS
 #7_DATA_BITS
 #8_DATA_BITS
 -> BC_PARITY
 #ODD_PARITY
 #EVEN_PARITY
 #NO_PARITY
 #FORCED_TO_0
 #FORCED_TO_1
 -> BC_DUPLEX_MODE
 #HALF_DUPLEX
 #FULL_DUPLEX
 -> BC_MODEM_TYPE
 ( numeric value )
 -> BC_LAYER2_ID
 ( numeric value )
 -> BC_L2_PROTOCOL
 #Q.921
 #X.25_LINK
 -> BC_LAYER3_ID
 ( numeric value )
 -> BC_L3_PROTOCOL
 #Q.931
 #X.25_PACKET

Multiple frame est., Octet 5b
not supported
 supported
Operation mode, Octet 5b
bit transparent mode
 protocol sensitive mode
LLI negotiation, Octet 5b
default, LLI = 256 only
full protocol negotiation
Assignor/Assignee, Octet 5b
message orig. is default assignee
message orig. is assignor only
In-band/out-band, Octet 5b
neg. is done with INFO messages
neg. is done with logical link zero
Number of stop bits, Octet 5c
not used
 1 bit
 1.5 bits
 2 bits
Number of data bits, Octet 5c
not used
 5 bits
 7 bits
 8 bits
Parity, Octet 5c
odd
even
 none
forced to 0
forced to 1
Duplex mode, Octet 5d
half duplex
 full duplex
Modem type, Octet 5d
range 0 through 63
Layer identifier, Octet 6
valid value: 2
Layer 2 protocol, Octet 6
Rec. Q.921 (I.441)
Rec. X.25, link level
Layer identifier, Octet 7
valid value: 3
Layer 3 protocol, Octet 7
Rec. Q.931 (I.451)
Rec. X.25, packet layer
Called Party Number IE (I#CALLED_NUM)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4

->CLDN_NUMBER_TYPE
   #UNKNOWN
   #INTERNATIONAL
   #NATIONAL
   #NETWORK_SPECIFIC
   #LOCAL_DIRECTORY
   #ABBREVIATED

->CLDN_NUMBERING_PLAN
   #UNKNOWN_PLAN
   #ISDN_PLAN
   #DATA_PLAN
   #TELEX_PLAN
   #NATIONAL_PLAN
   #PRIVATE_PLAN

->CLDN_NUMBER
   ( IA5 characters )

Type of number, Octet 3
unknown
international number
national number
network specific number
subscriber number
abbreviated number

Numbering plan, Octet 3
unknown
ISDN numbering plan Rec. E.164
data numbering plan Rec. X.121
telex numbering plan Rec. F.69
national standard numbering plan
private numbering plan

Number, Octet 4 *
max. length 32 octets

Called Party Subaddress IE (I#CALLED_SAD)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4

->CLDS_ADDRESS_TYPE
   #NSAP
   #USER_SPECIFIC

->CLDS_ODD/EVEN
   #EVEN_NUMBER
   #ODD_NUMBER

->CLDS_ADDRESS
   ( IA5 characters )

Type of subaddress, Octet 3
NSAP (X.213 / ISO 8348 AD2)
user specific

Odd/even indicator, Octet 3
even number of address digits
odd number of address digits

Subaddress, Octet 4 *
max. length 20 octets
Calling Party Number IE (I#CALLING_NUM)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_3A, OCTET_4

->CLGN_NUMBER_TYPE
   #UNKNOWN
   #INTERNATIONAL
   #NATIONAL
   #NETWORK_SPECIFIC
   #LOCAL_DIRECTORY
   #ABBREVIATED
   Type of number, Octet 3
   unknown
   international number
   national number
   network specific number
   subscriber number
   abbreviated number

->CLGN_NUMBERING_PLAN
   #UNKNOWN_PLAN
   #ISDN_PLAN
   #DATA_PLAN
   #TELEX_PLAN
   #NATIONAL_PLAN
   #PRIVATE_PLAN
   Numbering plan, Octet 3
   unknown
   ISDN numbering plan Rec. E.164
   data numbering plan Rec. X.121
   telex numbering plan Rec. F.69
   national standard numbering plan
   private numbering plan

->CLGN_PRESENTATION
   #PRESENT_ALLOWED
   #PRESENT_RESTRICTED
   #NUMBER_UNAVAIL
   Presentation ind., Octet 3a
   presentation allowed
   presentation restricted
   not available due to interworking

->CLGN.Screening
   #UNSCREENED
   #VERIFY_PASSED
   #VERIFY_FAILED
   #NETWORK_PROVIDED
   Screening indicator, Octet 3a
   user—provided, not screened
   user—provided, verified and passed
   user—provided, verified and failed
   network provided

->CLGN_NUMBER
   ( IA5 characters )
   max. length 32 octets

Calling Party Subaddress IE (I#CALLING_SAD)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4

->CLGS_ADDRESS_TYPE
   #NSAP
   #USER_SPECIFIC
   Type of subaddress, Octet 3
   NSAP (X.213 / ISO 8348 AD2)
   user specific

->CLGS_ODD/EVEN
   #EVEN_NUMBER
   #ODD_NUMBER
   Odd/even indicator, Octet 3
   even number of address digits
   odd number of address digits

->CLGS_ADDRESS
   ( IA5 characters )
   max. length 20 octets
**Call Identity IE (I#CALL_ID)**

Possible octet inclusions/exclusions:

OCTET_3

--> CL_CALL_ID  
  ( hex characters )

  Call identity, Octet 3 *  
  max. length 8 octets

**Call State IE (I#CALL_STATE)**

Possible octet inclusions/exclusions:

OCTET_3

--> CS_CODING_STANDARD  
  #CCITT  
  #INTERNATIONAL  
  #NATIONAL  
  #NETWORK_SPECIFIC  

--> CS_CALL_STATE  
  #NULL  
  #CALL_INIT  
  #OVERLAP_SENDING  
  #OUTGOING_CALL_PROC  
  #CALL_DELIVERED  
  #CALL_PRESENT  
  #CALL_RECEIVED  
  #CONNECT_REQUEST  
  #INCOMING_CALL_PROC  
  #ACTIVE  
  #DISC_REQUEST  
  #DISC_INDICATION  
  #SUSPEND_REQUEST  
  #RESUME_REQUEST  
  #RELEASE_REQUEST  
  #CALL_ABORT  
  #OVERLAP_RECEIVING  
  #RESTART_REQUEST  
  #RESTART  

Coding standard, Octet 3

  CCITT  
  other international standards  
  national standard  
  standard defined for the network  

Call state, Octet 3

  Null  
  Call Initiated  
  Overlap Sending  
  Outgoing Call Proceeding  
  Call Delivered  
  Call Present  
  Call Received  
  Connect Request  
  Incoming Call Proceeding  
  Active  
  Disconnect Request  
  Disconnect Indication  
  Suspend Request  
  Resume Request  
  Release Request  
  Call Abort  
  Overlap Receiving  
  Restart Request  
  Restart
Cause IE (I#CAUSE)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_3A, OCTET_4, OCTET_5

--> C_CODING_STANDARD
  #CCITT  Coding standard, Octet 3
  #INTERNATIONAL  CCITT
  #NATIONAL  other international standards
  #NETWORK_SPECIFIC  national standard
  standard defined for the network

--> C_LOCATION
  #USER  Location, Octet 3
  #LOCAL_PRIVATE  user
  #LOCAL_PUBLIC  private network serving local user
  #TRANSIT  public network serving local user
  #REMOTE_PUBLIC  transit network
  #REMOTE_PRIVATE  public network serving remote user
  #INTERNAT_NETWORK  private network serving remote user
  #BEYOND_INTERWORK  international network
  network beyond interworking point

--> C_RECOMMENDATION
  #REC_Q.931  Recommendation, Octet 3a
  #REC_X.21  Q.931
  #REC_X.25  X.21

--> C_CAUSE_VALUE
  #UNASSIGNED_NUMBER  Cause value, Octet 4
  #NO_ROUTE_TO_TRANSIT  Unassigned number
  #NO_ROUTE_TO_DEST  No route to transit network
  #CHANNEL_UNACCEPTABLE  No route to destination
  #CALL_AWARDED  Channel unacceptable
  #NORMAL_CLEARING  Call awarded
  #USER_BUSY  Normal call clearing
  #NO_USER_RESPOND  User busy
  #NO_ANSWER_FROM_USER  No user responding
  #CALL_REJECTED  No answer from user
  #NUMBER_CHANGED  Call rejected
  #NON_SELECTED_CLEARING  Number changed
  #NON_SELECTED_CLEARING  Non-selected user clearing
  #OUT_OF_SERVICE  Destination out of order
  #INVALID_NUMBER_FORMAT  #INVALID_NUMBER_FORMAT  Invalid number format
  #FACILITY_REJECTED  Facility rejected
  #STATUS_ENQ_RESPONSE  Response to STATUS ENQUIRY
  #NORMAL_UNSPECIFIED  Normal, unspecified
  #NO_CHANNEL_AVAIL  No circuit/channel available
  #OUT_OF_ORDER  Network out of order
  #TEMPORARY_FAILURE  Temporary failure
  #SWITCH_CONGESTION  Switching equipment congestion
  #ACCESS_INFO_DISCARD  Access information discarded
  #CIRCUIT_UNAVAIL  Requested circuit not available
  #RESOURCES_UNAVAIL_UNSPEC  Resources unavailable, unspecified
Channel Identification IE (I#CHANNEL_ID)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_3.1, OCTET_3.2, OCTET_3.3

->CID_INT_PRESENT
  #IMPLICIT
  #EXPLICIT

->CID_INT_TYPE
  #BASIC_INTERFACE
  #OTHER_INTERFACE

->CID_PREF/EXCL
  #PREFERRED
  #EXCLUSIVE

->CID_DCHANNEL
  #NOT_D_CHANNEL
  #D_CHANNEL

Interface ident., Octet 3
implicitly identified
explicitly identified

Interface type, Octet 3
basic interface
other interface

Preferred/Exclusive, Octet 3
preferred
exclusive

D–channel indicator, Octet 3
not D–channel
D–channel identified

Diagnostic(s), Octet 5 *
max. length 27 octets
CID_INFO_CHAN_SEL
#NO_CHANNEL
#AS_INDICATED
#ANY_CHANNEL
CID_INT_ID
( hex characters )
CID_CODING_STANDARD
#CCITT
#INTERNATIONAL
#NATIONAL
#NETWORK_SPECIFIC
CID_NUMBER/MAP
#NUMBER
#MAP
CID_CHANNEL/MAP_TYPE
#B_CHANNEL_UNITS
#H0_CHANNEL_UNITS
#H11_CHANNEL_UNITS
#H12_CHANNEL_UNITS
CID_MAP
( hex characters )
CID_NUMBER
( numeric value )
Info. chan. sel., Octet 3
no channel
as indicated
any channel
Interface ident., Octet 3.1 *
max. length 8 octets
Coding standard, Octet 3.2
CCITT
other international standards
national standard
standard defined for the network
Number/Map, Octet 3.2
number
map
Chan./Map type, Octet 3.2
B-channel units
H0-channel units
H11-channel units
H12-channel units
Slot map, Octet 3.3 *
max. length 4 octets
Channel number, Octet 3.3
range 0 through 127

Congestion Level IE (I#CONG_LEVEL)

CL_CONGESTION_LEVEL
#RECEIVER_READY
#RECEIVER_NOT_READY
Congestion level
receiver ready
receiver not ready

Date/time IE (I#DATE/TIME)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4, OCTET_5, OCTET_6, OCTET_7, OCTET_8

DT_YEAR
( numeric value )
Year, Octet 3
range 0 through 255
DT_MONTH
( numeric value )
Month, Octet 4
range 0 through 255
DT_DAY
( numeric value )
Day, Octet 5
range 0 through 255
DT.HOUR
( numeric value )
Hour, Octet 6
range 0 through 255
DT_MINUTE
( numeric value )
Minute, Octet 7
range 0 through 255
DT_SECOND
( numeric value )
Second, Octet 8
range 0 through 255
Display IE (I#DISPLAY)

Possible octet inclusions/exclusions:

OCTET_3

->D_DISPLAY
   (1A5 characters)

Display information, Octet 3
max. length 80 octets

End-to-end Transit Delay IE (I#END-END_DELAY)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_3A, OCTET_3B, OCTET_4, OCTET_4A, OCTET_4B, OCTET_5, OCTET_5A, OCTET_5B

->EE_CUMUL_DELAY
   (numeric value)

Cum. transit delay, Octet 3, 3a, & 3b
range 0 through 64536

->EE_REQ_DELAY
   (numeric value)

Req. end-end delay, Octet 4, 4a, & 4b
range 0 through 64536

->EE_MAX_DELAY
   (numeric value)

Max. end-end delay, Octet 5, 5a, & 5b
range 0 through 64536

Endpoint Identifier IE (I#ENDPOINT_ID)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4

->EP_USID
   (numeric value)

User service id., Octet 3
range 0 through 127

->EP_INTERPRETER
   #MATCHES_USID+TID
   #MATCHES_USID

Interpreter, Octet 4
matches USID and TID
matches USID and not TID

->EP_TID
   (numeric value)

Terminal identifier, Octet 4
range 0 through 63
Facility IE (I#FACILITY)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4, OCTET_5, OCTET_6

-->FC_SERVICE_DISCR
 #SUPP_SERVICES
 Service descr., Octet 3
 supplementary service applications
-->FC_CLASS
 #UNIVERSAL
 Class, Octet 4
 universal
 #APPLICATION-WIDE
 application-wide
 #CONTEXT-SPECIFIC
 context-specific
 #PRIVATE_USE
 private use
-->FC_FORM
 #PRIMITIVE
 Form, Octet 4
 primitive
 #CONSTRUCTOR
 constructor
-->FC_COMP_TAG
 #INVOKE
 Component tag, Octet 4
 invoke
 #RETURN_RESULT
 return result
 #RETURN_ERROR
 return error
 #REJECT
 reject
-->FC_LENGTH_FMT
 #ONE_OCTET
 Length format, Octet 5
 one octet
-->FC_COMP_LENGTH
 ( numeric value )
 Length of component, Octet 5
 range 0 through 127
-->FC_COMPONENT
 ( hex characters )
 Component, Octet 6 *
 max. length 32 octets

Feature Activation IE (I#FEAT_ACT)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_3A

-->FA_FEAT_NUM
 ( numeric value )
 Feature ID number, Octet 3 & 3a
 range 0 through 16384
Feature Indication IE (I#FEAT_IND)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_3A, OCTET_4

->FI_FEAT_NUM
    ( numeric value )
->FI_STATUS
    #DEACTIVATED
    #ACTIVATED
    #PROMPT
    #PENDING

Feature ID number, Octet 3 & 3a
    range 0 through 16384
Status indicator, Octet 4
    feature is in the deactivated state
    feature is in the active state
    feature prompt (waiting for input)
    feature is pending

High Layer Compatibility IE (I#HI_LAY_COMP)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4, OCTET_4A

->HL_CODING_STANDARD
    #CCITT
    #INTERNATIONAL
    #NATIONAL
    #NETWORK_SPECIFIC
->HL_INTERPRETATION
    #FIRST_CHARACTER
->HL_PRESENTATION
    #PROTOCOL_PROFILE
->HL_CHARACTER
    #TELEPHONY
    #FAX_GROUP_2/3
    #FAX_GROUP_4
    #MIXED_MODE
    #PROC_FORM
    #TELETEX
    #VIDEOTEX
    #TELEX
    #MHS
    #OSI_APPLICATION

Coding standard, Octet 3
    CCITT
    other international standards
    national standard
    standard defined for the network
Interpretation, Octet 3
    first HL characteristics to be used
Presentation method, Octet 3
    high layer protocol profile
HL characteristics, Octet 4
    telephony Rec. G.711
    fax group 2/3 Rec. T.62
    fax group 4 Rec. T.503
    mixed mode Rec. T.501
    processable form Rec. T.502
    teletex Rec. T.62 & T.70
    videotex Rec. T.504
    telex
    Message Handling Systems Rec. X.400
    OSI application Rec. X.200
Information Rate IE (I#INFO_RATE)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4, OCTET_5, OCTET_6

->IR_INCOMING

- 75BIT/S
- 150BIT/S
- 300BIT/S
- 600BIT/S
- 1200BIT/S
- 2400BIT/S
- 4800BIT/S
- 9600BIT/S
- 19200BIT/S
- 48000BIT/S

->IR_OUTGOING

- 75BIT/S
- 150BIT/S
- 300BIT/S
- 600BIT/S
- 1200BIT/S
- 2400BIT/S
- 4800BIT/S
- 9600BIT/S
- 19200BIT/S
- 48000BIT/S
Information Request IE (I#INFO_REQ)

Possible octet inclusions/exclusions:

OCTET_3

->IRQ_INDICATOR
    INFO_REQ_COMPL Info. request ind., Octet 3
    PROMPT_INFO information request completed
    prompt for additional information

->IRQ_INFO_TYPE
    UNDEFINED Type of information, Octet 3
    AUTH_CODE undefined
    ADDRESS_DIGITS authorization code
    TERMINAL_ID address digits
    terminal identification

Keypad IE (I#KEYPAD)

Possible octet inclusions/exclusions:

OCTET_3

->K_KEYPAD
    (1A5 characters) Keypad information, Octet 3
    * max. length 32 octets
Low Layer Compatibility IE (I#LOW_LAY_COMP)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_3A, OCTET_4, OCTET_4A, OCTET_4B, OCTET_5, OCTET_5A, OCTET_5B, OCTET_5C, OCTET_5D, OCTET_6, OCTET_6A, OCTET_7, OCTET_7A

->LL_CODING_STANDARD
   #Coding standard, Octet 3
   #CCITT
   #INTERNATIONAL
   #NATIONAL
   #NETWORK_SPECIFIC
->LL_TRANSFER_CAP
   #Info. trans. cap., Octet 3
   #SPEECH
   #UNRESTRICTED
   #RESTRICTED
   #3.1KHZ_AUDIO
   #7KHZ_AUDIO
   #VIDEO
->LL_NEG_IND
   #Negotiation ind., Octet 3a
   #NEG_NOT_POSSIBLE
   #NEG_POSSIBLE
->LL_TRANSFER_MODE
   #Transfer mode, Octet 4
   #CIRCUIT_MODE
   #PACKET_MODE
->LL_TRANSFER_RATE
   #Info. transfer rate, Octet 4
   #PACKET
   #64KBIT/S
   #2x64KBIT/S
   #384KBIT/S
   #1536KBIT/S
   #1920KBIT/S
->LL_STRUCTURE
   #Structure, Octet 4a
   #DEFAULT
   #8KHZ_INTEGRITY
   #SDU_INTEGRITY
   #UNSTRUCTURED
->LL_CONFIGURATION
   #Configuration, Octet 4a
   #POINT_TO_POINT
->LL_ESTABLISHMENT
   #Establishment, Octet 4a
   #DEMAND
->LL_SYMMETRY
   #Symmetry, Octet 4b
   #BIDIRECT_SYMMETRIC
->LL_TRANSFER_RATE_4B
   #Info. transfer rate, Octet 4b
   #PACKET
   #64KBIT/S
   #2x64KBIT/S
   #384KBIT/S
   #1536KBIT/S
   #1920KBIT/S
Layer identifier, Octet 5  
Valid value: 1  
Layer 1 protocol, Octet 5  
CCITT rate adaption V.110/X.30  
Rec. G.711 u-law  
Rec. G.711 A-law  
Rec. G.721 32 kbits/s ADPCM  
Rec. G.722 and G.724 7kHz audio  
Rec. G.7XX 384 kbit/s video  
Non-CCITT rate adaption  
CCITT rate adaption V.120  
CCITT rate adaption X.31 HDLC  
Sync/Async, Octet 5a  
synchronous  
asynchronous  
Negotiation, Octet 5a  
in-band negotiation not possible  
in-band negotiation possible  
User rate, Octet 5a  
indicated by E-bits Rec. I.460  
0.6 kbit/s Rec. V.6 and X.1  
1.2 kbit/s Rec. V.6  
2.4 kbit/s Rec. V.6 and X.1  
3.6 kbit/s Rec. V.6  
4.8 kbit/s Rec. V.6 and X.1  
7.2 kbit/s Rec. V.6  
8 kbit/s Rec I.460  
9.6 kbit/s Rec V.6 and X.1  
14.4 kbit/s Rec V.6  
16 kbit/s Rec I.460  
19.2 kbit/s Rec V.6  
32 kbit/s Rec I.460  
48 kbit/s Rec V.6 and X.1  
56 kbit/s Rec. V.6  
0.1345 kbit/s Rec. X.1  
0.100 kbit/s Rec. X.1  
0.075/1.2 kbit/s Rec. V.6 and X.1  
1.2/0.075 kbit/s Rec. V.6 and X.1  
0.050 kbit/s Rec. V.6 and X.1  
0.075 kbit/s Rec. V.6 and X.1  
0.110 kbit/s Rec. V.6 and X.1  
0.150 kbit/s Rec. V.6 and X.1  
0.200 kbit/s Rec. V.6 and X.1  
0.300 kbit/s Rec. V.6 and X.1  
12 kbit/s Rec. V.6  
Intermediate rate, Octet 5b  
not used  
8 kbit/s  
16 kbit/s  
32 kbit/s
NIC on Tx, Octet 5b
data not required
data required

NIC on Rx, Octet 5b
cannot accept data
can accept data

Flow Control on Tx, Octet 5b
not required
required

Flow Control on Rx, Octet 5b
cannot accept

Rate adaption, Octet 5b
header not included
header included

Multiple frame est., Octet 5b
not supported
supported

Operation mode, Octet 5b
bit transparent mode
protocol sensitive mode

LLI negotiation, Octet 5b
default, LLI = 256 only
full protocol negotiation

Assignor/Assignee, Octet 5b
message orig. is default assignee
message orig. is assignor only

In-band/out-band, Octet 5b
neg. is done with INFO messages
neg. is done with logical link zero

Number of stop bits, Octet 5c
not used
1 bit
1.5 bits
2 bits

Number of data bits, Octet 5c
not used
5 bits
7 bits
8 bits

Parity, Octet 5c
odd
even

Duplex mode, Octet 5d
half duplex
full duplex
---LL_MODEM_TYPE
    ( numeric value ) Modem type, Octet 5d
    range 0 through 63
---LL_LAYER2_ID
    ( numeric value ) Layer identifier, Octet 6
    valid value: 2
---LL_LAYER2_PROTOCOL
#ISO_1745 Layer 2 protocol, Octet 6
#Q.921 basic mode ISO 1745
#X.25_LINK Rec. Q.921 (I.441)
#X.25_MULTI Rec. X.25, link level
#EXT_LAPB Rec. X.25 Multilink
#HDLC_ARM extended LAPB (T.71)
#HDLC_NRM HDLC ARM (ISO 4335)
#HDLC_ABM HDLC NRM (ISO 4335)
#LAN_LLCP HDLC ABM (ISO 4335)
#X.75_SLP LAN LLC (ISO 8802/2)
#X.75_SLP Rec. X.75, SLP
---LL_LAYER2_INFO
    ( numeric value ) Optional layer 2, Octet 6a
    range 0 through 127
---LL_LAYER3_ID
    ( numeric value ) Layer identifier, Octet 7
    valid value: 3
---LL_LAYER3_PROTOCOL
#Q.931 Layer 3 protocol, Octet 7
#X.25_PACKET Rec. Q.931 (I.451)
#ISO_8208 Rec. X.25, packet layer
#ISO_8208 ISO 8208
#ISO_8348 ISO 8348
#ISO_8473 ISO 8473
#T.70 Rec. T.70, minimum network layer
---LL_LAYER3_INFO
    ( numeric value ) Optional layer 3, Octet 7a
    range 0 through 127

More Data IE (I#MORE_DATA)

NOTE
There are no selectors for this information element.
**Network Facilities IE (I#NETWORK_FACIL)**

Possible octet inclusions/exclusions:

OCTET_3, OCTET_3.1, OCTET_3.2, OCTET_4

- `->NF_NET_ID_LENGTH` (numeric value)  
  Network id. length, Octet 3, range 0 through 255
- `->NF_NET_ID_TYPE`  
  #USER_SPECIFIED_ID  
  #NAT_NET_ID  
  #INT_NET_ID  
  Network id. type, Octet 3.1  
  user specified  
  national network identification  
  international network ident.
- `->NF_NET_ID_PLAN`  
  #UNKNOWN  
  #CARRIER_ID_CODE  
  #DATA_ID_CODE  
  Network id. plan, Octet 3.1  
  unknown  
  Carrier Identification Code  
  Data network id. code (Rec. X.121)
- `->NF_NET_ID` (IA5 characters)  
  Network ident., Octet 3.2 *  
  max. length 16 octets
- `->NF_FACILITY_CODE` (hex characters)  
  Network facilities, Octet 4 *  
  max. length 32 octets

**Notification Indicator IE (I#NOTIFIC_IND)**

Possible octet inclusions/exclusions:

OCTET_3

- `->NL_DESCRIPTION`  
  #USER_SUSPENDED  
  #USER_RESUMED  
  #BEARER_CHANGE  
  Notification desc., Octet 3  
  user suspended  
  user resumed  
  bearer service change

**Packet Layer Binary Parameters IE (I#PKT_PARAMS)**

Possible octet inclusions/exclusions:

OCTET_3

- `->PP_FAST_SELECT`  
  #NOT_REQUESTED  
  #NO_RESTRICTIONS  
  #RESTRICTIONS  
  Fast select, Octet 3  
  not requested  
  requested with no restrictions  
  requested with restrictions
- `->PP_EXP_DATA`  
  #REQUEST_DENIED  
  #REQUEST_ACCEPTED  
  Expedited data, Octet 3  
  no request/request denied  
  request indicated/request accepted
Packet Size IE (I#PKT_SIZE)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4

-->PS_FORWARD
   ( numeric value )
   Forward value, Octet 3
   range 0 through 127

-->PS_BACKWARD
   ( numeric value )
   Backward value, Octet 4
   range 0 through 127

Packet Layer Window Size IE (I#PKT_WINDOW)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4

-->PW_FORWARD
   ( numeric value )
   Forward value, Octet 3
   range 0 through 127

-->PW_BACKWARD
   ( numeric value )
   Backward value, Octet 4
   range 0 through 127

Progress Indicator IE (I#PROGRESS_IND)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4

-->PL_CODING_STANDARD
    #CCITT
    #INTERNATIONAL
    #NATIONAL
    #NETWORK_SPECIFIC
    Coding standard, Octet 3
    CCITT
    other international standards
    national standard
    standard defined for the network
Redirecting Number IE (I#REDIRING_NUM)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_3A, OCTET_3B, OCTET_4

---> RDGN_NUMBER_TYPE
    #UNKNOWN
    #INTERNATIONAL
    #NATIONAL
    #NETWORK_SPECIFIC
    #LOCAL_DIRECTORY
    #ABBREVIATED

---> RDGN_NUMBERING_PLAN
    #UNKNOWN_PLAN
    #ISDN_PLAN
    #DATA_PLAN
    #TELEX_PLAN
    #NATIONAL_PLAN
    #PRIVATE_PLAN

---> RDGN_PRESENTATION
    #PRESENT_ALLOWED
    #PRESENT_RESTRICTED
    #NUMBER_UNAVAILABLE

---> RDGNSCREENING
    #UNSCREENED
    #VERIFY_PASSED
    #VERIFY_FAILED
    #NETWORK_PROVIDED

Type of number, Octet 3
unknown
international number
national number
network specific number
subscriber number
abbreviated number

Numbering plan, Octet 3
unknown
ISDN numbering plan Rec. E.164
data numbering plan Rec. X.121
telex numbering plan Rec. F.69
national standard numbering plan
private numbering plan

Presentation ind., Octet 3a
presentation allowed
presentation restricted
not available due to interworking

Screening indicator, Octet 3a
user—provided, not screened
user—provided, verified and passed
user—provided, verified and failed
network provided
->RDGN_REASON
  #CALL_FORWARD_BUSY
  #NO_REPLY
  #DTE_OUT_OF_ORDER
  #CALL_FWD_BY_DTE
  #UNCONDITIONAL

->RDGN_NUMBER
  ( IA5 characters )

Redirection reason, Octet 3b
  call forwarding or called DTE busy
  call forwarding no reply
  called DTE out of order
  call forwarding by the called DTE
  call forwarding unconditional

Number, Octet 4 *
  max. length 32 octets

---

Repeat Indicator IE (I#REPEAT_IND)

->RP_REPEAT_IND
  #PRIORITIZED

Repeat indicator
  prioritized list

---

Restart Indicator IE (I#RESTART_IND)

Possible octet inclusions/exclusions:

OCTET_3

->RI_CLASS
  #INDICATED_CHANNEL
  #SINGLE_INTERFACE
  #ALL_INTERFACES

Class, Octet 3
  indicated channels
  single interface
  all interfaces

---

Segmented Message IE (I#SEGMENT)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4

->SG_FIRST_IND
  #SUBSEQUENT
  #FIRST_SEGMENT

First segment ind., Octet 3
  subsequent segment to first
  first segment

->SG_REMAINING
  ( numeric value )

Segments remaining, Octet 3
  range 0 through 127
->SG_MESSAGE_TYPE
#ALERT
#CALL_PROC
#PROG
#SETUP
#CONN
#SETUP_ACK
#CONN_ACK
#USER_INFO
#SUSP_REJ
#RES_REJ
#HOLD
#SUSP
#RES
#HOLD_ACK
#SUSP_ACK
#RES_ACK
#HOLD_REJ
#RET
#RET_ACK
#RET_REJ
#DISC
#REST
#REL
#REST_ACK
#REL_COM
#SEGMENT
#FAC
#REG
#NOTIFY
#STATUS_ENQ
#CON_CON
#INFO
#STATUS

Message type, Octet 4
Alerting
Call proceeding
Progress
Setup
Connect
Setup acknowledge
Connect acknowledge
User information
Suspend reject
Resume reject
Hold
Suspend
Resume
Hold acknowledge
Suspend acknowledge
Resume acknowledge
Hold reject
Retrieve
Retrieve acknowledge
Retrieve reject
Disconnect
Restart
Release
Restart acknowledge
Release complete
Segment
Facility
Register
Notify
Status enquiry
Congestion control
Information
Status

Sending Complete IE (I#SEND_COMP)

\[ NOTE \]
There are no selectors for this information element.
Shift IE (I#SHIFT)

->SH_TYPE

#LOCKING
#NON_LOCKING

->SH_CODESET

#CODESET0
#CODESET5
#CODESET6
#CODESET7

Shift type
locking
non-locking

Codeset ident.
I.451 (Q.931) IE
national use IE
local network specific IE
user specific IE

Signal IE (I#SIGNAL)

Possible octet inclusions/exclusions:

OCTET_3

->SI_VALUE

#DIAL_ON
#RING_BACK_ON
#INTERCEPT_ON
#CONGESTION_ON
#BUSY_ON
#CONFIRM_ON
#ANSWER_ON
#CALL_WAITING_ON
#OFF_HOOK_ON
#TONES_OFF
#ALERTING_ON_0
#ALERTING_ON_1
#ALERTING_ON_2
#ALERTING_ON_3
#ALERTING_ON_4
#ALERTING_ON_5
#ALERTING_ON_6
#ALERTING_ON_7
#ALERTING_OFF

Signal value, Octet 3
dial tone on
ring back tone on
intercept tone on
network congestion tone on
busy tone on
confirm tone on
answer tone on
call waiting tone on
off-hook warning tone on
tones off
alerting on – pattern 0
alerting on – pattern 1
alerting on – pattern 2
alerting on – pattern 3
alerting on – pattern 4
alerting on – pattern 5
alerting on – pattern 6
alerting on – pattern 7
alerting off

Service Profile Id. IE (I#SPID)

Possible octet inclusions/exclusions:

OCTET_3

->SPID

(IA5 characters)

SPID, Octet 3 *
max. length 30 octets
Switchhook IE (I#SWITCHHOOK)

Possible octet inclusions/exclusions:

OCTET_3

->SW_VALUE #ON_HOOK #OFF_HOOK

Switchhook value, Octet 3

on-hook

off-hook

Transit Delay Selection IE (I#TRANS_DEL_SEL)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_3A, OCTET_3B

->TDS_VALUE ( numeric value )

->TDS_VALUE

( numeric value )

->TDS_VALUE

( numeric value )

Transit delay, Octet 3

range 0 through 3

Transit delay, Octet 3a

range 0 through 127

Transit delay, Octet 3b

range 0 through 127

Transit Network Select IE (I#TRANS_NW_SEL)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4

->TR_NET_ID_TYPE

#USER_SPECIFIED_ID

#NAT_NET_ID

#INT_NET_ID

->TR_NET_ID_PLAN

#UNKNOWN

#CARRIER_ID_CODE

#DATA_ID_CODE

->TR_NET_ID ( IA5 characters )

Network id. type, Octet 3

user specified

national network identification

international network ident.

Network id. plan, Octet 3

unknown

Carrier Identification Code

Data network id. code (Rec. X.121)

Network ident., Octet 4 *

max. length 16 octets
User-user Information IE (I#UU_INFO)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4

->UU_PROTOCOL_DISCR
   ( numeric value )
 Protocol discr., Octet 3
 range 0 through 255

->UU_USER_INFO
   ( hex characters )
 User information, Octet 4 *
 max. length 254 octets
The ISDN D–Channel Emulation program has the functionality of the ISDN D–Channel Monitor and layer 2 emulation and layer 3 simulation capabilities.

**Layer 2 Emulation** The ISDN D–Channel Emulation can maintain up to eight different links simultaneously. A link is uniquely identified by its SAPI (service access point identifier) and its TEI (terminal endpoint identifier). The layer 2 emulation maintains this information for each of the eight links along with other link specific information (e.g. timer durations, data field lengths, window size, retransmission counters, modulus mode, state variables, and the layer 2 state). The emulation determines the applicable link when a layer 2 event occurs and selects the appropriate set of parameters.

**Layer 3 Simulation** The ISDN D–Channel Simulation can maintain up to eight simultaneous connections. A connection is uniquely identified by its call reference value. The layer 3 simulation maintains information relevant to the connection in separate data structures for each of the eight connections.

The data structure contains all the different parameters used to construct information elements for a connection. It also holds information relevant to the connection (i.e. information element parameter values, channel types, call states, etc.).

The test manager supports layer 3 connection multiplexing. When a message is received, the call reference value identifies the connection, selects the appropriate data structure, and starts the test manager. If a new call reference value is received, the active test manager state is saved at the current state number and the test manager is restarted using the new data structure.

The previously saved test manager state resumes where it left off if a message with the original call reference value is received again. Consequently, eight different connections can be managed by the test manager, using a separate data structure for each test script.
15.1 Live Data

Data is passed to the monitor decode and then through to the triggers or the test manager. Based on the trigger selection, data is passed through to the filters. The filters permit or restrict the flow of data into the capture RAM, disk, or display.

The test manager, if active, processes the received data and generates a response. The test manager can utilize the services of the message builder in order to generate a layer 3 response.

Finally, outgoing data is framed by the layer 2 emulation and sent out to the physical interface.

Figure 15-1 ISDN Emulation Data Flow Diagram – Live Data

Display topic

Live Data function key

MONITOR ( -- )
Selects the live data mode of operation. All incoming events and transmitted frames are decoded and displayed in real-time.
15.2 Playback

Data can be played back from either capture RAM or disk without interfering with an active test (i.e. dropping the link) as shown in Figure 15-2.

---

**Figure 15-2 ISDN Emulation Data Flow Diagram — Offline Processing**

- FROM_CAPT HALT
  - Display topic
  - Playback RAM function key

- FROM_DISK HALT PLAYBACK
  - Display topic
  - Playback Disk function key

**HALT (--)**

Selects the playback mode of operation. Data is retrieved from capture RAM or a disk file, decoded, and then displayed or printed. Capture to RAM is suspended in this mode.
15.3 Simultaneous Live Data and Playback

Live data can be recorded to disk while playing back data from capture RAM.

**Figure 15-3 ISDN Emulation Data Flow Diagram — Freeze Mode**

- **FROM_CAPT FREEZE**
  - **Capture topic**
  - **Record to Disk** function key
  - **Display topic**
  - **Playback RAM** function key

**FREEZE ( --- )**

Enables data to be recorded to disk while data from capture RAM is played back.
IDACOM testers can emulate D-Channel data on a WAN, Basic Rate, or Primary Rate interface. This section describes configuration commands for layers 1, 2, and 3.

The tester can emulate a user (TE) or network (NT).

**USER_EMUL** ( -- )
Selects user emulation.

**NTWK_EMUL** ( -- )
Selects network emulation.

**NOTE**
For basic rate, user and network are selected on the Home processor.

**USR*NET** ( -- address )
Contains the emulation state. Valid values are:

- **TE** Terminal emulation
- **NT** Network emulation

Example:
Change from a user emulation to a network emulation.

```
USR*NET @
TE = ( Obtain current configuration )
IF
   NTWK_EMUL
   ( Change to a network emulation )
ENDIF
```
16.1 Layer 1

The section describes commands to configure the physical layer for each interface type.

Basic Rate

The Basic Rate interface is configured on the Home processor prior to loading the D-Channel Emulation application. However, commands exist within the emulation for interface configuration.

Each application processor is associated with a serial port which has a port identifier. This port identifier is used in most of the Basic Rate layer 1 configuration commands.

PORT_BRID ( -- port id )
  Returns the port identifier for the Basic Rate D-Channel port (Port A for BRA/BRA tester).

PORT_BRID2 ( -- port id )
  Returns the port identifier for the Basic Rate D-Channel Port B (BRA/BRA tester).

The S/T bus can be configured for point-to-point operation (one TE device attached) or point to multipoint operation (up to eight TE devices attached).

PPMP ( port\connect mode -- )
  Where: connect mode = P2P point-to-point mode (default)
  P2MP point to multipoint

  Establishes the bus configuration of the unit.

  Example 1:
  Configure the S/T bus for point-to-point operation.
  PORT_BRID P2P PPMP

  Example 2:
  Configure the S/T bus for Port B of a BRA/BRA tester for point to multipoint operation.
  PORT_BRID2 P2MP PPMP

When a phone is connected to the tester, the voice encoding method is either A-law or µ-law.

CODE_TYPE ( port\voice code -- )
  Where: voice code = U-LAW µ-law voice encoding (default)
  A-LAW A-law voice encoding

  Selects the type of voice encoding.

  Example:
  Select µ-law voice encoding.
  PORT_BRID U-LAW CODE_TYPE
The voice CODEC can be set to generate tones over the voice channel.

**TONEN_TYPE** ( port\tone -- )
Where:  tone = AUD_RING

<table>
<thead>
<tr>
<th>SPECIAL_AR</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>440Hz/480Hz</td>
</tr>
<tr>
<td></td>
<td>Repeat tones 1 second on, 3 seconds off.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BUSY</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Special Audible Ring</td>
</tr>
<tr>
<td></td>
<td>Tones on for 1 second, followed by single 440 Hz tone on for 200 milliseconds, then off for 3 seconds. Repeat.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BUSY_VFY</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Busy Verify</td>
</tr>
<tr>
<td></td>
<td>One burst tone 1.75 seconds before attendant intrudes, followed by burst of 650 ms on 8 to 20 seconds apart as long as the call lasts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONFIRM</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Confirm</td>
</tr>
<tr>
<td></td>
<td>350Hz/440Hz</td>
</tr>
<tr>
<td></td>
<td>Burst on 100 ms. Burst off 100 ms. Repeat three times.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CALL_WAIT</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Call wait</td>
</tr>
<tr>
<td></td>
<td>440Hz/OFF</td>
</tr>
<tr>
<td></td>
<td>One burst for 250 ms.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DIAL</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dial</td>
</tr>
<tr>
<td></td>
<td>350Hz/440Hz</td>
</tr>
<tr>
<td></td>
<td>Steady tone.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXEC_OVD</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exec. Override</td>
</tr>
<tr>
<td></td>
<td>440Hz/OFF</td>
</tr>
<tr>
<td></td>
<td>One burst tone 3 seconds before override occurs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INTERCEPT</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Intercept</td>
</tr>
<tr>
<td></td>
<td>440Hz/620Hz</td>
</tr>
<tr>
<td></td>
<td>First tone on 250 ms followed by second tone on for 250 ms. Repeat.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RECALL_DIAL</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall Dial</td>
</tr>
<tr>
<td></td>
<td>350Hz/440Hz</td>
</tr>
<tr>
<td></td>
<td>On 100 ms. Off 100 ms. Repeat three times, followed by a steady tone.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REORDER</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reorder</td>
</tr>
<tr>
<td></td>
<td>480Hz/620Hz</td>
</tr>
<tr>
<td></td>
<td>Tone on 250 ms. Tone off 250 ms. Repeat.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OFF</th>
<th>Audible Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Turn tone off</td>
</tr>
<tr>
<td></td>
<td>Turns repetitive tones, such as a dial tone, off.</td>
</tr>
</tbody>
</table>

Selects the tone generated over the voice channel.

Example:
Configure the tester so that a voice channel on B2-Channel is connected to the CODEC. Tones are sent over the voice channel.

```
PORT_BRID BCHAN2 VOICE BCHAN_SRC  ( Connect B2-Channel to the voice codec )
PORT_BRID U-LAW CODE_TYPE  ( Select mu law voice encoding )
PORT_BRID DRT_SERIAL DROUTING DROP  ( Place the B-Channel on line )
PORT_BRID DIAL TONE_TYPE  ( Send a dial tone )
```
When emulating a network, the terminal device might require power to be supplied from the network.

**SET_PS** ( port/state -- )

Where: state = PS1_OFF  
PS1+  
PS1-  
PS2_OFF  
PS2+  
PS2-  

Sets the power supply to the indicated state.

**GET_PS** ( port -- power supply 1 state\power supply 2 state )

Where: state = PS1_OFF  
PS1+  
PS1-  
PS1_ERROR  
PS2_OFF  
PS2+  
PS2-  
PS2_ERROR  

Returns the state of the two power supplies.

Example:
Configure the tester for point-to-point, forward power, and network emulation to power supply 1.

PORT_BRID P2P_PPMP  
PORT_BRID PS1+ SET_PS  
PORT_BRID GET_PS DROP  
PS1_ERROR =  
IF  
T." Error - Check other ends power supply." TCR  
ENDIF  
NTWK_EMUL  

(ISDN Programmer's Manual)
The B-Channel data stream can be protocol data routed to another application processor for analysis, voice channel routed to the external telephone connector, or protocol data directed to the external port.

**BCHAN_SRC** ( port\source\channel -- )

Where: source =
- BCHAN1 (B1-Channel)
- BCHAN2 (B2-Channel)
- PP1 (AP #1 / AP #4 for Port B of BRA/BRA)
- PP2 (AP #2 / AP #5 for Port B of BRA/BRA)
- EXT_1 (external connector 1)
- EXT_2 (external connector 2)
- VOICE (codec)

channel =
- BCHAN1 (B1-Channel)
- BCHAN2 (B2-Channel)

Selects the source of transmitted data for the specified B-Channel. The BCHAN_SRC command can perform single and cross channel loopback, as well as routing data to the application processors (see the examples following DROUTING command in this section).

**DROUTING** ( port\route -- flag )

Where: route =
- DRT_OFF - turn off the flow of data to the application processors.
- DRT_SERIAL - route B-Channels to the application processors.

Connects the B-Channels to an application processor and returns true if successful.

Example 1:
Configure the tester to connect the B-Channels through to the application processors.

1 SEQ[
   PORT_BRID PP1 BCHAN1 BCHAN_SRC
   PORT_BRID PP2 BCHAN2 BCHAN_SRC
   PORT_BRID DRT_SERIAL DROUTING
   IF
      " B-Channels are connected to application processors."
   ELSE
      " Bus not activated. B-Channels are not connected."
   ENDIF
   W.NOTICE
]SEQ
Example 2:
Configure the tester with B1-Channel in self loopback mode (see Figure 16-1).

PORT_BRID BCHAN1 DUP BCHAN_SRC  ( Put B1-Channel in self loopback )
PORT_BRID DRT_SERIAL DROUTING  ( Enable flow of data to B-Channels )
IF
   " B-Channels are looped back."
ELSE
   " Bus not activated. B-Channels are not looped back."
ENDIF
W.NOTICE

Figure 16-1 Selfloop Configuration Diagram
Example 3:
Create a sequence that crossloops the B-Channels (i.e. connects B1-Channel to B2-Channel and vice versa). See Figure 16-2.

```
3 SEQ{
    PORT_BRID BCHAN1 BCHAN2 BCHAN_SRC
    PORT_BRID BCHAN2 BCHAN1 BCHAN_SRC
    PORT_BRID DRT_SERIAL DROUTING
    IF
        " B- Channels are cross connected."
    ELSE
        " Bus not activated. B-Channels are not cross connected."
    ENDIF
    W.NOTICE
}
```

![Crossloop Configuration Diagram](image)

**Figure 16-2 Crossloop Configuration Diagram**
Example 4:
Configure the tester to connect B1–Channel to External #1 and B2–Channel to voice (see Figure 16–3).

```
PORT_BRID EXT_1 BCHAN1 BCHAN_SRC ( Route B1 to external 1 )
PORT_BRID VOICE BCHAN2 BCHAN_SRC ( Route B2 to voice )
PORT_BRID DRT_SERIAL DROUTING ( Enable flow of data to B-Channel )
IF
  " B1 routed to external #1. B2 routed to voice."
ELSE
  " Bus not activated."
ENDIF
W.NOTICE
```

![Configuration Diagram](image)

Figure 16–3  B1 to External 1, B2 to Voice Configuration Diagram

Example 5:
Create a sequence that disconnects the B–Channels.

```
4 SEQ{
  PORT_BRID DRT_OFF DROUTING
  IF
    " B–Channels disconnected."
  ELSE
    " Bus not activated. B–Channels are already disconnected."
  ENDIF
  W.NOTICE
}SEQ
```
The S/T bus can be activated/deactivated in the emulation.

**L1_ACTIVATE (--)**
Activates the bus.

- **Emulation topic**
  - Activate function key

**L1_DEACTIVATE (--)**
Deactivates the bus.

- **Emulation topic**
  - Deactivate function key

### Primary Rate
The Primary Rate interface is configured on the Home processor prior to loading the D-Channel Emulation application. Corresponding configuration commands do not exist in the application.

### WAN
The WAN interface is configured on the application processor after loading the D-Channel Emulation and switching to the application processor.

![Emulation Configuration Menu](image)

**Physical Layer:**
- **Emulation Interface** TO DCE
- **Interface Type** V.35
- **Interframe Fill** FLAG

**Timer Duration:**
- **Idle Link (T203)** 300

**Flags:**
- **XID Negotiate Proc** OFF
- **RR Polling Action** OFF

**Bit Rate** 64000
**External Tx Clock** OFF

**Protocol Emulation:**
- L.2 State Machine ON

**Special SAPI:**
- Packet Communication 16

**Figure 16-4 Emulation Configuration Menu (WAN)**
INTERFACE_WAKEUP ( -- )

Configures the physical interface.

*NOTE*
Use INTERFACE_WAKEUP once after all physical layer changes are made.

Physical Layer:

*Emulation Interface*

Selects the physical type of emulation and determines whether the tester generates clocking or expects to receive clocking, as well as setting which pins transmit and receive data.

*NOTE*
Refer to Table 16-1 for clocking selections depending on the emulation interface.

INTERFACE ( -- address )

Contains the physical interface type. Valid values are:

0 to DCE connector  Clocking must be supplied by the attached equipment.
1 to DTE connector  Tester supplies all clocking information to the interface connection.

Example:

Select the 'to DTE' interface.

```
1 INTERFACE !
INTERFACE_WAKEUP
```

*Interface Type*

IF=V28 ( -- )

Selects the V.28/RS-232C connector and electrically isolates the other connectors on the port.

<RS232C/V.28 function key>

IF=V11 ( -- )

Selects the V.11/X.21 connector and electrically isolates the other connectors on the port.

<RS422/V.11 function key>

IF=V35 ( -- )

Selects the V.35 connector (default) and electrically isolates the other connectors on the port.

<V.35 function key>

IF=V36 ( -- )

Selects the V.36 (RS-449) connector and electrically isolates the other connectors on the port.

<RS449/V.36 function key>

*NOTE*

A WAN tester has a V.28, V.11, and either a V.35 or V.36 connector. These commands are only applicable if the program is running on a WAN interface.
→ Interframe Fill
Selects the bit pattern transmitted between blocks of data.

**INTERFRAME-FILL** (-- address)
Contains the interframe fill character.

Examples:
Set interframe fill to MARK.
MRK INTERFRAME-FILL !

 маршрут MARK function key

Set interframe fill to FLAG (default).
SYNC INTERFRAME-FILL !

 маршрут FLAG function key

→ Bit Rate
The interface speed can be selected from preset values on the Interface Port Speed Menu or set to a user-defined speed. The bit rate of the ISDN line can also be measured. The following table lists the associated action depending on the emulation interface and clocking selections.

<table>
<thead>
<tr>
<th>Clocking</th>
<th>TO DTE</th>
<th>TO DCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>Select</td>
<td>Measure</td>
</tr>
<tr>
<td>ON</td>
<td>Measure</td>
<td>Select</td>
</tr>
</tbody>
</table>

Table 16-1 Effect of Clocking and Emulation Interface Selections on Bit Rate

⚠️ **NOTE**
*Clocking is provided by the attached equipment when the bit rate can be selected.*

**INTERFACE-SPEED** (-- address)
Contains the current bit rate (default is 64000).

 маршрут Modify Value function key
Enter Interface Port Speed (50 to 128000):

⚠️ **NOTE**
*Integer values must be written to INTERFACE-SPEED. Thus, to obtain a bit rate of 134.5, either 134 or 135 can be written to INTERFACE-SPEED.*

→ External Tx Clock
Enables/disables external clocking.

**IF-CLOCK** (-- address)
Contains the state of the external clock. Valid values are:

- YES enable external clock
- NO disable external clock
16.2 Layer 2

This section describes the various commands and variables used for layer 2 configuration. Layer 2 configuration is identical for all interface types.

The ISDN D-Channel Emulation supports automatic emulation of up to eight layer 2 data links. Layer 2 configuration is divided into link independent and dependent commands. The following parameters are link independent and affect the operation of the layer 2 emulation as a whole.

![Emulation Configuration Menu (Basic Rate)](image)

**Timer Duration:**
- **Idle Link (T203)**
  The T203 timer starts after a frame has been transmitted and stops when a frame is received. The duration of the T203 timer is the maximum time on the link without exchange of frames.

  **T203-DUR#** (--- value)
  Returns the value, in tenths of seconds, of the T203 timer.

  **T203 (value ---)**
  Sets the value, in tenths of seconds, of the T203 timer.

  **Example:**
  If the value of T203 is less than 30 seconds, set to 30 seconds.

  ```
  T203-DUR# 300 <
  IF 300 T203 ENDIF
  ```

  **Modify Value** function key
EMULATION CONFIGURATION

Flags:

→ XID Negotiate Proc
Establishes common values for the data field length (N201), the window size, and the retransmission timer (T200) between two peers.

XID_ON ( -- )
Enables the XID negotiation procedure.

 quizzes Enabled function key

XID_OFF ( -- )
Disables the XID negotiation procedure.

 quizzes Off function key

?XID_ON ( -- flag )
Returns true if the XID negotiation procedure is enabled.

Example:
Disable XID negotiation if XID negotiation is enabled.

?XID_ON
IF
   XID_OFF
ENDIF

→ RR Polling Action
Detects faulty data link connections. Receiver ready frames are transmitted when the T203 timer expires. If receiver ready frames are in turn received, then the link is up.

RR_POLL_ON ( -- )
Enables the receiver ready polling action.

 quizzes Enabled function key

RR_POLL_OFF ( -- )
Disables the receiver ready polling action.

 quizzes Off function key

?RR_POLL_ON ( -- flag )
Returns true if the receiver ready polling action is enabled.
Protocol Emulation:
→ L.2 State Machine
The following commands control the operation of the layer 2 data link state machine and the user management layer or the network assignment source procedure. The layer 2 state machine can be turned off to send a response to a layer 2 received frame.

L2_ON ( -- )
Enables the layer 2 data link state machine.

L2_OFF ( -- )
Disables the layer 2 data link state machine.

?L2_ON ( -- flag )
Returns true if the layer 2 data link state machine is enabled.

ML_ON ( -- )
Enables the operation of the user management layer or the network assignment source procedure layer.

ML_OFF ( -- )
Disables the operation of the user management layer or the network assignment source procedure layer.

?ML_ON ( -- flag )
Returns true if the user management layer or the network assignment source procedure layer is enabled.

Example:
Enable layer 2 data link state machine, and user management layers.

L2_ON
ML_ON

ON function key

Special SAPI:
→ Packet Communication
Decodes layer 3 information within layer 2 I or UI frames with a specified SAPI value according to the X.25 (1980/1984) protocol.

NOTE
This SAPI value cannot be changed within test scripts. Use the Emulation Configuration Menu to modify this SAPI.

DATA COM—SAPI# ( -- value )
Returns the value of the data communications (packet data) SAPI (default value is 16).
Link Setup

The following commands and variables apply only to the selected link.

![Figure 16-6 Link Setup Menu](image)

- **Link No**
  Specifies a particular link CES (connection endpoint suffix).

- **CES** (-- address)
  Contains the current CES. Valid values are 0 through 7.

- **SA** (value --)
  Selects a particular link CES. Valid values are 0 through 7.

  ![Modify Value function key](image)

- **DLCI Value**
  - **SAPI** (address)
    Sets the SAPI (service access point identifier) for the selected link.
  - **SAPI** (address)
    Contains the SAPI for the currently selected link. Standard values include:

    | Value | Description                      |
    |-------|----------------------------------|
    | 0     | Used for call setup signaling.   |
    | 16    | Used for D-Channel packet communication (X.25). |
    | 63    | Used for management / ASP layer.  |
SAPI (value --)
Sets the SAPI for the selected link.

Example:
Set the SAPI for the current link to 63.
63 SAPI

Modify Value function key

→ TEI
Sets the TEI (terminal endpoint identifier) for the selected link.

VTEI* (-- address)
Contains the TEI for the currently selected link.

TEI or FTEI (value --)
Sets the current TEI without affecting the layer 2 state machine. Valid values are 0 through 127. The layer 2 state machine is not affected. Default values include:

127 Basic Rate (TEI not yet assigned).
0-7 Primary Rate and WAN.

Example:
Set the TEI for the current link to 1.
1 TEI

Modify Value function key

ATEI (value --)
Assigns the current TEI to 'value'. For the user, both the layer 2 and management layer states are changed to the TEI assigned state (state 4). For the network, the layer 2 state is changed to the TEI assigned state (state 4). Valid values are 0 through 126.

Modulus:

Mode
Determines whether normal (modulo 8) or extended (modulo 128) frame operation is used.

MODE (value --)
Where: value = 0 - normal (modulo 8)
      1 - extended (modulo 128)

Sets the modulus operation mode.

MODE-FLG* (-- address)
Contains the current setting of the modulus mode. 0 indicates normal mode (default) and 1 indicates extended mode.

MODULUS* (-- address)
Contains the modulus value for the currently selected link. Valid values are 8 or 128.
Link TEI:

The link TEI mode can be either manual or automatic. Manual mode is used, for example, when testing equipment has a 'hard-wired' TEI. When the link TEI mode is set to manual, the TEI request procedure is disabled and the TEI value in VTEI* is set.

PREF-TEI* ( -- address )

Contains the current setting of the link TEI mode. 0 indicates automatic mode (default) and 1 indicates manual mode. After setting the preferred TEI setting, the MDL_ASSIGN_R command (see Section 17) should be executed to indicate to the management layer entity that the specified TEI should be associated with the current link. This link will then be in state 4 (TEI assigned).

Timer Duration:

Primary (T200)

T200-DUR* ( -- address )

Contains the current value, in tenths of seconds, for the T200 timer (default is 1 second).

T200 ( value -- )

Specifies the time, in tenths of seconds, between SABM/E retransmissions during link setup. It also sets the value of the T201 timers equal to that of the T200 timer and sets the T202 timer to twice that of the T200 timer (default is 1 second).

Example:
If the T200 timer is set for less than 2 seconds, set to 2 seconds.

T200-DUR* @ 20 <
IF
  20 T200
ENDIF

Modify Value function key

The T200 timer can be set to a preferred value without setting values in the T201 and T202 timers.

T200-DUR# ( -- value )

Returns the preferred value, in tenths of seconds, of the T200 timer (default is 1 second).

PT-T200 ( value -- )

Sets the preferred value, in tenths of seconds, of the T200 timer.

Example:
If the preferred value of the T200 timer is less than 1 second, set to 1 second.

T200-DUR# 10 <
IF
  10 PT-T200
ENDIF
→ **T200-RX (XID Neg)**
Specifies the maximum length of a received I or UI frame for the XID negotiate procedure.

**T200-RX** (address)
Contains the value, in tenths of seconds, for the T200 timer during XID negotiation (default is 1 second).

**T200RX** (value)
Sets the value, in tenths of seconds, of T200 timer for XID negotiation (default is 1 second).

Example:
When the T200 timer for XID negotiation is less than 1 second, set to 1 second.

T200-RX* @ 10 <
IF
 10 T200RX
ENDIF

Modify Value function key

**Max. Retransmission:**
→ **Primary Event (N200)**
The maximum number of times that a frame is retransmitted after the expiry of the T200 timer is determined by N200.

**N200** (address)
Contains the current setting of N200 (default value is 3).

**N200** (value)
Sets the value of N200.

Modify Value function key

**Data Field Length:**
→ **N201 (XID Neg)**
N201 represents the maximum number of octets in an I frame information field.

**N201** (address)
Contains the value for N201 (default is 260).

**N201** (value)
Sets the value of N201. Valid values are 0 through 286.

Example:
If N201 does not equal 260, set to 260.

N201* @ 260 = 0=
IF
 260 N201
ENDIF

Modify Value function key
N201# ( -- value )
  Returns the preferred maximum receive packet size.

PT–N201 ( value -- )
  Sets the preferred maximum receive packet size (default is 260).
  
  Example:
  When the preferred maximum receive packet size is less than 260, set to 260.

  N201# 260 <
  IF
    260 PT–N201
  ENDIF

→ N201–TX (XID Neg)
  Specifies the maximum length of the information field in transmitted I or UI frames.

N201–TX* ( -- address )
  Contains the value of N201 for transmit frames (default is 260).

N201TX ( value -- )
  Sets the value of N201 for transmit frames.

  Modify Value function key

N201 can be set to a preferred value for outgoing frames.

N201–TX# ( -- value )
  Returns the preferred maximum outgoing packet size.

PT–N201TX ( value -- )
  Sets the preferred maximum outgoing packet size.

Window:
→ K
  Specifies the maximum number of I frames that can be transmitted before a response is received.

K* ( -- address )
  Contains the current setting of the window size (default is 7).

K ( value -- )
  Sets the window size.

  Modify Value function key
The window size can be set to a preferred value.

K# ( -- value )
    Returns the preferred value of the window size.

PT-K ( value -- )
    Sets the preferred value of the window size.

→ K-RX
    Specifies the requested value of k (transmit window) for the peer. Used during XID negotiate procedures.

K-RX* ( -- address )
    Contains the current setting of the window size received from the peer following an XID negotiation (default value is 7).

KRX ( value -- )
    Sets the preferred value of K transmitted in an XID frame.

Modify Value function key

MAX* ( -- address )
    Contains the maximum number of outstanding I frames.

W ( value -- )
    Sets the window parameters. Sets K to equal 'value' and sets MAX* to 'value - 1'.

Management Layer/Assignment Source Procedure Setup

The management layer/assignment source procedure setup specifies the parameters used during TEI assignment and XID negotiation.

Management Layer Setup

<table>
<thead>
<tr>
<th>Timer Duration:</th>
<th>ML/ASP Retransmission:</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ TEI Request (T202) 20</td>
<td>TEI Request (N202) 3</td>
</tr>
<tr>
<td>XID Negotiate (TM20) 10</td>
<td>XID Negotiate (NM20) 3</td>
</tr>
<tr>
<td></td>
<td>ID Denied (N204) 2</td>
</tr>
</tbody>
</table>

Figure 16-7 Management Layer Setup Menu
Timer Duration:

- \textit{TEI Request (T202)}
  Specifies the minimum time between retransmission of the TEI identity request messages. The value of T202 should be set to \(4 \times T200\) seconds.

\texttt{T202-DUR# ( -- value )}
Returns the value, in tenths of seconds, of T202 (default is 2 seconds).

\texttt{T202 ( value -- )}
Sets the value, in tenths of seconds, of T202.

Example:
If T202 is less than 2 seconds, set to 2 seconds.

\begin{verbatim}
T202-DUR# 20 <
IF
  20 T202
ENDIF
\end{verbatim}

- \textit{XID Negotiate (TM20)}
  Specifies the response time of a peer to an XID frame. If a response is not received prior to the expiry of TM20, the XID frame is retransmitted.

\texttt{TM20-DUR# ( -- value )}
Returns the value, in tenths of seconds, of TM20 (default is 1 second).

\texttt{TM20 ( value -- )}
Sets the duration, in tenths of seconds, of the XID procedure timer.

ML/ASP Retransmission:

- \textit{TEI Request (N202)}
  Specifies the maximum number of TEI assignment requests retransmissions.

\texttt{N202* ( -- address )}
Contains the value of N202 (default is 3).

\texttt{N202 ( value -- )}
Sets the value of N202.

- \textit{XID Negotiate (NM20)}
  Specifies the maximum number of XID command frame retransmissions.

\texttt{NM20* ( -- address )}
Contains the value of NM20.

\texttt{NM20 ( value -- )}
Sets the value of NM20 (default is 3).
→ ID Denied (N204)
Specifies the maximum number of times that a user attempts to acquire a TEI due to ID denials from the network.

N204* ( -- address )
Contains the value of N204.

N204 ( value -- )
Sets the value of N204 (default is 2).

![Assignment Source Procedure](image)

Figure 16-8 Assignment Source Procedure Menu

Timer Duration:
→ TEI ID Check (T201)
Specifies the minimum time between network retransmission of the TEI identity check message.

T201-DUR# ( -- value )
Returns the value, in tenths of seconds, of the T201 timer (default is 1 second).

T201 ( value -- )
Sets the value, in tenths of seconds, of the T201 timer (default is 1 second).

⚠️ NOTE
See the Management Layer Setup Menu for descriptions of the XID Negotiate items.
This section describes commands used during layer 2 emulation.

Section 17.1 describes commands that can be used inside or outside of the state machine. These commands change the state of the state machine or modify the outgoing frame appropriately.

Section 17.2 describes the in-state commands, such as state machine control, layer 2 services, and the in-state send commands. These commands are used under the control of the state machine.

Section 17.3 describes the out-of-state commands. These commands can be used in any state that the state machine can be in. A number of commands are provided as support to the out-of-state words. These commands can change the fields inside the outgoing frame, as well as changing the state of the machine. In essence, it is possible to change any parameter of the outgoing frame as needed.

Section 17.4 describes transmit mode commands, i.e. continuous or repetitive information frame transmission.

17.1 State-Independent Commands

The following commands can be used inside or outside of the state machine. In either case, they change the outgoing frame appropriately or change the state of the state machine.

**BROADCAST ( -- )**

Sets the layer 2 emulation for broadcast mode. UI frames use a TEI of 127.

**PT_TO_PT ( -- )**

Sets the layer 2 emulation to use the current TEI for transmitted UI frames.
Command/Response Bits

CMND* ( -- address )
Contains the value of the command bit (0 or 1).

RESP* ( -- address )
Contains the value of the response bit (0 or 1).

State Variables

State variables are used to keep track of frames transmitted between peers with sequence numbers cycling through the number of frames (as set by the MODE command, see Section 16).

VS ( value -- )
Specifies the V(S) (send state variable) count identifying the sequence number of the next information frame transmitted by the tester. Valid values are 0 through \( n - 1 \), where \( n \) is the modulus number.

VA ( value -- )
Specifies the V(A) (acknowledge state variable) count identifying the sequence number of the last acknowledged frame. Valid values are 0 through \( n - 1 \), where \( n \) is the modulus number.
VA* ( --address )
Contains the current V(A).

Send Link Setup Menu
→ VA

VR ( value -- )
Specifies the V(R) (receive state variable) count identifying the sequence number of the next in sequence information frame expected to be received. Valid values are 0 through n – 1, where n is the modulus number.

Send Link Setup Menu
→ VR

VR* ( --address )
Contains the current V(R).

Send Link Setup Menu
→ VR

FRMR Bits
The frame reject response indicates to the peer that it is unable to recover from an error in transmission. The error cannot be recovered by merely retransmitting the frame. The reason for rejecting the frame is carried in four bits: W, X, Y, and Z bits.

The W bit, when set to 1, indicates that the received control field was undefined or not implemented. The bad control field is returned in octets 5 and 6 of the FRMR frame.

The X bit, when set to 1, indicates that the received control field was invalid because a received information field was not valid with that frame, or the frame was a supervisory or unnumbered frame with an invalid length.

The Y bit, when set to 1, indicates that the length of the received information field exceeded the length set in N201.

The Z bit, when set to 1, indicates that the N(R) value was invalid.

W0 ( -- )
Sets the W bit to 0.

Send Link Setup Menu
→ W Bit (0)

W1 ( -- )
Sets the W bit to 1.

Send Link Setup Menu
→ W Bit (1)
FRMRW* (—address )
Contains the value of the W bit.

X0 (— )
Sets the X bit to 0.

X1 (— )
Sets the X bit to 1.

FRMRX* (—address )
Contains the value of the X bit.

Y0 (— )
Sets the Y bit to 0.

Y1 (— )
Sets the Y bit to 1.

FRMRY* (—address )
Contains the value of the Y bit.

Z0 (— )
Sets the Z bit to 0.

Z1 (— )
Sets the Z bit to 1.
Example:
Send a frame reject indicating that the received information field exceeded the maximum field length (specified by N201).

\[ \text{W0} \]
\[ \text{X0} \]
\[ \text{Y1} \]
\[ \text{Z0} \]

**FRMR**

---

**Retransmission Counters**

The retransmission counters are used by the state machine to count the number of times that a frame has been retransmitted.

\[ \text{NOTE} \]
*The state machine clears the counters when it begins retransmission. However, a counter can be set to a desired value inside a test script.*

**RC (value --)**
Specifies the RC (retransmission counter) identifying the number of times a particular poll sequence has been transmitted to the peer.

\[ \text{Send Link Setup Menu} \]
\[ \rightarrow \text{Retransmission (RC)} \]

Example:
Without changing the value of N200, change the retransmission count of SABME's sent to a non-responding peer.

\[ \text{SABME} \quad (\text{sends SABMEs}) \]

After the first SABME is transmitted, type:

\[ 1 \text{ RC} \]

This has the effect of only sending one retransmission of the SABME.

**RC* ( -- address )**
Contains the retransmission counter value.

\[ \text{Send Link Setup Menu} \]
\[ \rightarrow \text{Retransmission (RC)} \]

**PRC (value --)**
Specifies the PRC (poll/response counter) identifying the current number of outstanding polls.

\[ \text{Send Link Setup Menu} \]
\[ \rightarrow \text{Poll/Response (PRC)} \]
APC ( value -- )
Sets the advance poll counter. The advance poll counter specifies the current number of unacknowledged outstanding polls transmitted in advance of an information frame.

APC* ( -- address )
Contains the advance poll counter value.

RC20 ( value -- )
Sets the value of the XID retransmission counter.

RC20* ( -- address )
Contains the XID retransmission counter value.

RC201 ( value -- )
Sets the value of the 201 retransmission counter. This counter is used only by the network assignment source procedure during an identification check request sequence.

RC201* ( -- address )
Contains the value of the 201 retransmission counter.

RC-ML1* ( -- address )
Contains the number of TEI assignment requests.

RC-ML2* ( -- address )
Contains the number of attempts to acquire a TEI value, due to ID denials from the network.

Miscellaneous

XIDV ( N201 TX\N201\K\T200 -- )
Sets the XID variables N201-TX*, N201*, K-RX*, and T200-RX* respectively. These values are used in the next XID frame that is transmitted.

Link Setup Menu
→ N201-TX (XID Neg)
→ N201 (XID Neg)
→ K-RX
→ T200-RX (XID Neg)

Example:
Send an XID frame indicating a field length of 260 for both the transmitter and receiver, 7 for the transmit window size, and a 1 second duration for the retransmission timer.

260 ( N201 TX )
260 ( N201 )
7 ( K )
10 ( T200 in tenths of seconds )
XIDV ( Defines XID variables)
XIDC ( Sends the XID command frame )
NRVALID* ( -- address )
Contains a flag which is true if the incoming frame has a valid N(R).

K—CCITT* ( -- address )
Contains the CCITT default value of K.

17.2 In-State Commands

The following commands are executed under control of the layer 2 state machine. All frames sent as a result of these commands use values for NR, NS, P, F, W, X, Y, and Z that are determined totally by the state machine.

State Machine Control

The following commands change the state of the application.

L2ST ( value -- )
Sets the data link layer state of the currently selected link. Refer to Table 17–1 for valid layer 2 state values.

Send Link Setup Menu
→ L2 State

STATE—L2* ( --address )
Contains the layer 2 state for the currently selected link.

Send Link Setup Menu
→ L2 State

MLST ( value -- )
Sets the state of the user management layer for the currently selected link. Refer to Table 17–2 for the valid management state values.

STATE—ML* ( --address )
Contains the management layer state for the currently selected link.

ASPST ( value -- )
Sets the state of the network assignment source procedure. Refer to Table 17–3 for valid ASP state values.

STATE—ASP* ( --address )
Contains the assignment source procedure state for the currently selected link.
### Layer 2 States

<table>
<thead>
<tr>
<th>State</th>
<th>CCITT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>TEI unassigned</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>TEI being assigned</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Waiting for establishment</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>TEI assigned</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Release waiting</td>
</tr>
</tbody>
</table>

**Normal States**

<table>
<thead>
<tr>
<th>State</th>
<th>CCITT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>7.0</td>
<td>Normal</td>
</tr>
<tr>
<td>71</td>
<td>7.1</td>
<td>Reject</td>
</tr>
<tr>
<td>72</td>
<td>7.2</td>
<td>Own receiver busy</td>
</tr>
<tr>
<td>73</td>
<td>7.3</td>
<td>Own receiver busy and reject</td>
</tr>
<tr>
<td>74</td>
<td>7.4</td>
<td>Peer receiver busy</td>
</tr>
<tr>
<td>75</td>
<td>7.5</td>
<td>Peer receiver busy and reject</td>
</tr>
<tr>
<td>76</td>
<td>7.6</td>
<td>Peer receiver and own receiver busy</td>
</tr>
<tr>
<td>77</td>
<td>7.7</td>
<td>Peer receiver and own receiver busy and reject</td>
</tr>
</tbody>
</table>

**Timer Recovery States**

<table>
<thead>
<tr>
<th>State</th>
<th>CCITT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>8.0</td>
<td>Normal</td>
</tr>
<tr>
<td>81</td>
<td>8.1</td>
<td>Reject</td>
</tr>
<tr>
<td>82</td>
<td>8.2</td>
<td>Own receiver busy</td>
</tr>
<tr>
<td>83</td>
<td>8.3</td>
<td>Own receiver busy and reject</td>
</tr>
<tr>
<td>84</td>
<td>8.4</td>
<td>Peer receiver busy</td>
</tr>
<tr>
<td>85</td>
<td>8.5</td>
<td>Peer receiver busy and reject</td>
</tr>
<tr>
<td>86</td>
<td>8.6</td>
<td>Peer receiver and own receiver busy</td>
</tr>
<tr>
<td>87</td>
<td>8.7</td>
<td>Peer receiver and own receiver busy and reject</td>
</tr>
</tbody>
</table>

### Table 17–1 Layer 2 States

### Management States

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TEI unassigned</td>
</tr>
<tr>
<td>2</td>
<td>TEI being assigned</td>
</tr>
<tr>
<td>3</td>
<td>XID response waiting</td>
</tr>
<tr>
<td>4</td>
<td>TEI assigned</td>
</tr>
</tbody>
</table>

### Table 17–2 Management States
Services

The layer 2 services are primitive operations provided between the data link layer and the management layer.

Primitives prefixed with 'DL' are used for communication between the data link layer and layer 3.

Primitives prefixed with 'MDL' are used for communication between the management layer and the data link layer.

These commands are executed under the control of the layer 2 state machine.

**DL_ESTABLISH ( -- )**
Establishes a layer 2 connection. If the bus is not activated (BRA only), then DL_ESTABLISH activates the bus. If a TEI is not already assigned, then the TEI assignment procedure is executed (user side only) and if the XID (exchange identification) process is enabled, then it is executed. An SABM or SABME is sent and the emulation waits for a UA response.

**DL_RELEASE ( -- )**
Releases a layer 2 connection. A DISC (disconnect) frame is sent. When a UA frame is received, the link goes into the TEI assigned state.

**MDL_ASSIGN_R ( TEI -- )**
Associates the current link with the given TEI.

**NOTE**
This command is used only in states 1 or 3 for the network, 2 or 3 for the user.

**MDL_ASSIGN_I ( -- )**
Used by the data link layer to indicate to the layer management entity, the need for a TEI to be associated with the CES specified in the primitive message unit.
MDL_REMOVE ( -- )

Used by the layer management to request that the data link layer remove the association between the current TEI value with the specified CES. The sub-address specifies the data link from which the TEI is to be removed.

RESET_BUSY ( -- )

If the layer 2 state machine is in an 'own receiver busy' state, then issuing this command causes the emulation to send out a RR frame. The layer 2 emulation then changes to a state to accept information frames.

If the layer 2 state machine is not in an 'own receiver busy' state, then no action occurs.

Services topic
  Unbusy function key

MAKE_BUSY ( -- )

The emulation sends out a RNR frame, with the poll bit set to 1 if the layer 2 state machine is not in an 'own receiver busy' state. The layer 2 state machine then goes into an 'own receiver busy' state.

If the peer sends a RR frame, the layer 2 state machine responds with a RNR frame.

If the layer state machine is already in an 'own receiver busy' state, then no action occurs.

Services topic
  Busy function key

?OWN_BUSY ( -- flag )

Returns true if the layer 2 state machine is in one of the 'own receiver busy' states.

Send Commands

The following commands operate under the control of the layer 2 state machine. Consequently, the sent frames use values for the C/R bit, the P/F bit, N(R), N(S), W, X, Y, and Z that are determined by the state machine. The current values of these variables prior to executing the commands are ignored, even though they might have been changed. The transmitted SAPI and TEI values are the current values.

SABM ( -- )

Transmits a SABM (set asynchronous balance mode) frame. The emulation expects a UA frame in response, otherwise it will transmit the SABM frame N200 times. After receiving a UA frame, the emulation will be in the information transfer state (state 7.0).

Sending a SABM sets the MODE flag to 0 (normal mode).

Send topic
  SABM function key
SABME (---)
Transmits a SABME (set asynchronous balance mode extended) frame. The emulation expects a UA frame in response, otherwise it will retransmit the SABME frame N200 times. After receiving a UA frame, the emulation will be in the information transfer state.

Transmitting a SABME sets the MODE flag to 1 (extended mode).

Send topic
SABME function key

UA (---)
Transmits a UA (unnumbered acknowledge) frame.

Send topic
UA function key

RRC (---)
Transmits an RR (receiver ready frame) with the command bit set.

Send topic
RR function key

RNRC (---)
Transmits a RNR (receiver not ready) frame with the command bit set.

Send topic
RNR function key

REJC (---)
Transmits a REJ (reject) frame with the command bit set.

Send topic
REJ function key

DISC (---)
Transmits a DISC (disconnect) frame. The emulation expects a UA response. When the UA frame is received, the emulation will go into the disconnect mode.

Send topic
DISC function key

DM (---)
Transmits a DM (disconnected mode) frame. This indicates to the peer that the data link layer is in a state where multiframe operation cannot be performed.

Send topic
DM function key
FRMR (--)
Transmits an FRMR (frame reject) frame. The values for the W, X, Y, and Z bits are taken from the values set by W0, W1, X0, X1, etc.

Send topic
FRMR function key

XIDC (--) 
Initiates the XID (exchange identification) procedure and transmits an XID frame with the parameters N201 transmit, N201 receive, K, and T200 set to the preferred values.

Send topic
XID function key

SEND_I (address\length --)
Transmits a single information frame.

Example:
Transmit a layer 3 ALERT message.

SEND_UI (address\length --)
Transmits an unnumbered layer 3 I frame.

SEND_MUI (address\length --)
Transmits an unnumbered management layer/assignment source procedure I frame. The SAPI value used is 63.
Commands Using Data Buffers

Data buffers can only be used with the DL_DATA, DL_UDATA or MDL_UDATA in–state commands or with the SENDF, S:DATA, S:UDATA or S:MUDATA out–of–state commands (see Section 17.3). The data buffers hold user specified data.

PKT ( value -- )
  Specifies the buffer used as the data source for a send command. Valid buffers are 1 through 8. If the specified buffer is 0, keyboard entry is used.

PKT? ( -- value )
  Returns the buffer used as the data source for a send command. A 0 indicates that keyboard entry is used.

MD1, MD2, MD3, MD4, MD5, MD6, MD7, MD8 ( address-- )
  Sets the contents of the buffers 1 through 8 to the specified string. See the example following DL_DATA.

DL_DATA ( address -- ) or ( -- )
  Transmits a layer 3 packet using acknowledged operation.

This command expects either a string address or nothing on the stack. If PKT has been set to a buffer number, the contents of the buffer is sent. If PKT is set to 0, a string address must be specified.

Example:
Send a layer 3 call proceeding packet.

```
X'08010202' MD1
1 PKT DL_DATA
```

![Figure 17-2 Call Proceeding Message](image)
17.3 Out-of-State Commands

The following commands send frames outside the layer 2 state machine.

These commands use the existing values of SAPI, TEI, NR, NS, P, F, W, X, Y, and Z as needed. Since these values can be changed, the frames differ from those sent under the control of the layer 2 state machine.

Sequence Numbers

Sequence numbers are state variable numbers that are transmitted inside the frame. N(R) is the expected send sequence number of the next received information frame. N(S) is the send sequence number of transmitted information frames.

NR (value --)
Sets the value of N(R). Valid values are 0 through n - 1, where n is the modulus number (8 or 128).

MNR* (-- address)
Contains the value of N(R).

NS (value --)
Sets the value of N(S).

MNS* (-- address)
Contains the value of N(S).
P/F Bit

In command frames, the P/F bit is referred to as the P (poll) bit; in response frames, it is referred to as the F (final) bit. The P bit is set to 1 in order to solicit a response from the peer. The F bit set to 1 indicates that the frame is being sent in response to a poll.

F1 (---)
Sets the F bit to 1.

Control Field Setup Menu
→ F Bit (1)

F0 (---)
Sets the F bit to 0.

Control Field Setup Menu
→ F Bit (0)

P1 (---)
Sets the P bit to 1.

Control Field Setup Menu
→ P Bit (1)

P0 (---)
Sets the P bit to 0.

Control Field Setup Menu
→ P Bit (0)

Send Commands

SENF (address ---) or (---)
Transmits a user-defined string as a layer 2 frame. The entire frame, including the header, is defined inside the string. SENDF uses the data buffers, as described in Section 17.2.

Example:
Transmit a SABME using the SENDF command.
X"00817F" MD1
1 PKT
SENF

S:SABME (---)
Transmits a SABME (set asynchronous balanced mode extended) command frame.

S:SABM (---)
Transmits a SABM (set asynchronous balance mode) command frame.
S:UA ( -- )
Transmits a UA (unnumbered acknowledge) command frame.

S:RRC ( -- )
Transmits an RR (receiver ready) command frame.

S:RR ( -- )
Transmits an RR (receiver ready) response frame.

S:RNRC ( -- )
Transmits an RNR (receiver not ready) command frame.

S:RNR ( -- )
Transmits an RNR (receiver not ready) response frame.

S:REJC ( -- )
Transmits an REJ (reject) command.

S:REJ ( -- )
Transmits an REJ (reject) response.

S:DISC ( -- )
Transmits a DISC (disconnect) command.

S:DM ( -- )
Transmits a DM (disconnected mode) response.

S:FRMR ( -- )
Transmits an FRMR (frame reject) response.

S:XIDC ( -- )
Transmits an XID (exchange identification) command.

S:XID ( -- )
Transmits an XID (exchange identification) response.

S:XID0 ( -- )
Transmits an XID (exchange identification) response with an information field length of zero.

S:XID4 ( -- )
Transmits an XID (exchange identification) response with the first four header bytes of the XID packet.

S:XIDC0 ( -- )
Transmits an XID (exchange identification) command with an information field length of zero.

S:XIDC4 ( -- )
Transmits an XID (exchange identification) command with the first four header bytes of the XID packet.

S:I ( address\length -- )
Transmits an I (information) frame. The layer 3 data is specified by the given address and length.
S:UI ( address\length -- )
    Transmits a UI (unnumbered information) frame.

S:MUI ( address\length -- )
    Transmits a UI (unnumbered information) frame, but with a SAPI of 63 for layer 2 management procedures.

\NOTE
    The SEND_I, SEND_UI, SEND_MUI, S:I, S:UI, and S:MUI send commands are usually used in conjunction with a layer 3 data source command. See Section 18.1 for a description of the data source commands.

S:DATA ( address -- ) or ( -- )
    Transmits a layer 3 packet in an I frame using acknowledged operation. S:DATA uses the data buffers as described in Section 17.2.

S:UDATA ( address -- ) or ( -- )
    Transmits a layer 3 packet in a UI frame using unacknowledged operation. S:UDATA uses the data buffers as described in Section 17.2.

S:MUDATA ( address -- ) or ( -- )
    Transmits a management layer (or assignment source procedure) packet in a UI frame with SAPI 63 using unacknowledged operation. S:MUDATA uses the data buffers as described in Section 17.2.

### 17.4 Transmit Mode

PKTS ( value -- )
    Specifies the number of times to transmit an information frame.

\Send Data Source Menu
    \Queuing Procedure
        Repetitive function key

CONT_ON ( -- )
    Sets the continuous information frame transmit mode.

\Send Data Source Menu
    \Queuing Procedure
        Continuous function key

CONT_OFF ( -- )
    Turns off the continuous information frame transmit mode.
17.5 Frame Errors

Frames and messages can be sent with correct or incorrect CRC's (FCS), or can be aborted during transmission.

CRC_ERROR ( -- )
Transmits the next frame or message with a CRC error. Subsequent frames and messages will be sent correctly.

DO_ABORT ( -- )
Aborts the next transmitted frame or message. Subsequent frames and messages will be sent correctly.

GOOD_CRC ( -- )
Transmits the next frame or message correctly.
The ISDN D–Channel layer 3 simulation is used to create either correct or incorrect layer 3 messages. The sequence of these messages can be controlled by received events, keyboard commands, timers, other programs, etc. Simulation differs from emulation in that emulation provides a full and correct implementation of either the network or user side of the protocol. The emulation does not send any incorrect or out of sequence messages.

The simulation environment provides functions which, when used in conjunction with the test manager, give a very powerful and flexible base for simulation of ISDN D–Channel signalling. This section presents an overview of the commands that can be used to provide ISDN simulation.

Simulation of the D–Channel layer 3 protocol can be implemented in various degrees of complexity depending on the requirement and experience of the user.

### 18.1 Layer 3 Data Source

Several data sources are used for layer 3 send commands. Transmitted data can be a pre–defined raw data packet, a layer 3 default message or the contents of a message pool buffer.

**RAW ( type -- address\length )**

Where: type = 1 consecutive integers
2 all 1s (hex FF)
3 all 0s (hex 00)
4 alternating 1s and 0s (hex 55)

Returns the address and length of a buffer containing data. This raw data can subsequently be sent inside an information frame.

Send Data Source Menu

- Data Source
- Predefined Packets function key

**BYTES ( value -- )**

Sets the number of bytes (maximum is 260) to be transmitted in a raw information frame.

Example:
Transmit a layer 3 packet, 20 bytes long, containing all 1s.

```
20 BYTES               ( Length of I frame )
2 RAW                  ( Select data type )
SEND_I                ( Send the I frame )
```
POOL (string -- address \ length)

Returns the address and length of the specified pool buffer. See Section 18.5 for a description of managing pool buffers.

Example:
"SETUP#1" POOL SEND_UI

MESG (message identifier -- address \ length)

Returns the address and length of a pre-defined layer 3 message of the specified message type. See Section 18.4 for a description of the message builder.

Example:
M#REL MESG SEND_I

<table>
<thead>
<tr>
<th>Command Type</th>
<th>Frame Type</th>
<th>In-State</th>
<th>Out-of-State</th>
</tr>
</thead>
<tbody>
<tr>
<td>General (addr \ count --)</td>
<td>I</td>
<td>SEND_I</td>
<td>S:I</td>
</tr>
<tr>
<td></td>
<td>UI</td>
<td>SEND_UI</td>
<td>S:UI</td>
</tr>
<tr>
<td></td>
<td>MUI</td>
<td>SEND_MUI</td>
<td>S:MUI</td>
</tr>
<tr>
<td>Command/MD buffer (string --) or (--)</td>
<td>I</td>
<td>DL_DATA</td>
<td>S:DATA</td>
</tr>
<tr>
<td></td>
<td>UI</td>
<td>DL_UDATA</td>
<td>S:UDATA</td>
</tr>
<tr>
<td></td>
<td>MUI</td>
<td>MDL_UDATA</td>
<td>S:MUDATA</td>
</tr>
<tr>
<td>Message (message id --)</td>
<td>I</td>
<td>MESG SEND_I</td>
<td>MESG S:I</td>
</tr>
<tr>
<td></td>
<td>UI</td>
<td>MESG SEND_UI</td>
<td>MESG S:UI</td>
</tr>
<tr>
<td></td>
<td>MUI</td>
<td>MESG SEND_MUI</td>
<td>MESG S:MUI</td>
</tr>
<tr>
<td>Pool (string --)</td>
<td>I</td>
<td>POOL SEND_I</td>
<td>POOL S:I</td>
</tr>
<tr>
<td></td>
<td>UI</td>
<td>POOL SEND_UI</td>
<td>POOL S:UI</td>
</tr>
<tr>
<td></td>
<td>MUI</td>
<td>POOL SEND_MUI</td>
<td>POOL S:MUI</td>
</tr>
<tr>
<td>Raw (n --)</td>
<td>I</td>
<td>RAW SEND_I</td>
<td>RAW S:I</td>
</tr>
<tr>
<td></td>
<td>UI</td>
<td>RAW SEND_UI</td>
<td>RAW S:UI</td>
</tr>
<tr>
<td></td>
<td>MUI</td>
<td>RAW SEND_MUI</td>
<td>RAW S:MUI</td>
</tr>
</tbody>
</table>

Table 18-1 Send Commands

The data source buffer can be modified before sending to use a specific call reference value and flag.

ALTER_CR (address \ length -- address \ length)

Modifies the call reference in the specified layer 3 message.

DEST_SIDE (--)

Sets the call reference flag of subsequent messages modified with ALTER_CR to 1 (destination side).

ORIG_SIDE (--)

Sets the call reference flag of subsequent messages modified with ALTER_CR to 0 (origination side).

RX-CALLREF (-- address)

Contains the call reference value used when the message is sent from the call destination side.
TX—CALLREF ( -- )
Contains the call reference value used when the message is sent from the call origination side.

Example:
Use a call reference value of 10 for a connect message sent from the call origination side.
10 TX—CALLREF !
ORIG_SIDE
M#CONN MESG ALTER_CR SEND_I

18.2 Connection Management

Layer 3 simulation supports signalling procedures for up to eight simultaneous connections (CN’s). Connections are distinguished from each other by their call reference value. When a layer 3 message is received, the call reference is compared against all connections. If matched, the data structure for that connection is made ‘active’, and the data it contains is used in generating the simulator’s response to the event.

There are eight identical data structures which contain data for each connection. At any one time, only one of these connections (and hence only one of the data structures) is ‘active’, and any reading or writing of connection-related data is done using the active data structure.

#CN0, #CN1, #CN2, #CN3, #CN4, #CN5, #CN6, #CN7 ( -- value )
Identifies the connection.

=CN ( CN -- )
Sets the active connection.

Example:
#CN3 =CN ( Makes connection 3 the current connection )

?CN ( -- currently active CN )
Returns the currently active connection.

CN0, CN1, CN2, CN3, CN4, CN5, CN6, CN7 ( -- )
Sets the active connection.
The following commands are used to manipulate the data for each connection.

**CLEAR\_CN ( -- )**
Clears the data structure associated with the current connection. The data structure can then be reused for a new connection.

**CLEAR\_ALL\_CNS ( -- )**
Resets all data for all connections.

⚠️ **WARNING**
*Use this command with caution.*

**CN\_DEALLOC ( -- )**
Flags the current connection as being unused.

**?CN\_FREE ( -- flag )**
Returns true if the current connection is unused.

**?CN\_ALLOC ( -- CN\_1 ) or ( -- 0 )**
Searches for an unused connection. If found, the connection is cleared, flagged as being in use, and the connection number and a true flag are returned. If no unused connections exist, a false flag is returned.

**COPY\_CN ( src CN\_dest CN -- )**
Copies the connection data from one connection to another.

**CN\_INIT ( flag\_SAPI -- )**
Where: flag = #ORIG origination side
       #DEST destination side

Sets the SAPI, call reference flag, and call reference value for the current connection. In addition, this command sets the default information element field values, and generates default parameter values for all information elements for the current connection.

If the call reference flag is specified as the destination side, then the call reference value for this connection is automatically set to the last decoded message. The call reference value is taken from $MSG\_CRVALUE.

If the call reference flag is specified as the origination side, then the call reference value for this connection can be set using the SELECT\_CR command.
Connection Related Data Management

The following data is kept for each connection:
- An 'in use' flag.
- The current test manager state number.
- The channel type in use for the connection (B or D-Channel).
- The channel number in use for the connection (only valid when channel type is B).
- The national and international protocol discriminators.
- The SAPI (service access point identifier).
- The CES (connection endpoint suffix).
- The call reference flag.
- The call reference value.
- A set of all the information element parameters used in coding layer 3 messages.
- A set of pre-coded IE's.

=CN_TM_STATE ( state -- )
?CN_TM_STATE ( -- state )
Sets or reads the state of the connection test manager state.

\v
See Section 20.6 for a description of how the layer 3 connection multiplexer manages the connection test manager state.

=CN_CHANNEL_TYPE ( type -- )
?CN_CHANNEL_TYPE ( -- type )
Where: type = #D_CHAN D-Channel
       #B_CHAN B-Channel
Sets or reads the channel type of the current connection.

\v
Setting or clearing the channel type does not affect connection management. Rather, these commands provide a convenient location to store and access the channel type.

Example:
Specify that the channel being used for the current connection is a D-Channel.
#D_CHAN =CN_CHANNEL_TYPE

=CN_CHANNEL_NUM ( number -- )
?CN_CHANNEL_NUM ( -- number )
Sets or reads the channel number.

\v
Setting or clearing the channel number does not affect connection management. Rather, these commands provide a convenient location to store and access the channel number.
=CN_INT_PD ( pd -- )
?CN_INT_PD ( -- pd )
Sets or reads the international protocol discriminator. If set, this value will be used for all international (single octet message type) messages built with the MESSAGE> and MESG commands. The default protocol discriminator for each message type will be used if this value is not set.

=CN_NAT_PD ( pd -- )
?CN_NAT_PD ( -- pd )
Sets or reads the national protocol discriminator. If set, this value will be used for all national (two octet message type) messages built with the MESSAGE> and MESG commands. The default protocol discriminator for each message type will be used if this value is not set.

=CN_CEI ( SAPI\CES -- )
?CN_CEI ( -- SAPI\CES )
Sets or reads the SAPI and CES for the current connection.

---

Call Reference Management

=CN_CR_FLAG ( flag -- )
?CN_CR_FLAG ( -- flag )
Where: flag = #ORIG origination side
     #DEST destination side

Sets or reads the connection call reference flag. The call reference flag identifies the end of the logical link that originated the call. This value will be used for all messages built with the MESSAGE> and MESG commands.

=CN_CR_VALUE ( value -- )
?CN_CR_VALUE ( -- value )
Sets or reads the connection call reference value. Each layer 3 connection is associated with a call reference value, used in the header of layer 3 signalling messages. This value will be used for all messages built with the MESSAGE> and MESG commands.

=CR_LENGTH ( length -- )
?CR_LENGTH ( -- length )
Sets or reads the call reference length.

⚠️ NOTE
"The call reference length setting applies to all connections. It is not possible to have a separate call reference length for each connection."

CR_VAL_MAX ( -- value )
Returns the maximum call reference value based on the current call reference length.

SELECT_CR ( -- value )
Returns a call reference value between 1 and CR_VAL_MAX which is not currently in use for an outgoing call. 0 is returned if none are available.
CLEAR_CR ( -- )
Sets the call reference value for the active connection to #CREF_UNUSED, indicating that the call reference has been released.

---

Information Element Management

A set of all parameters used to construct information elements is kept for each connection. These parameters are used when response messages are created by the tester. In addition, a set of buffers is kept for each connection for storing generated IE’s.

COPY_PARAMS ( source CN\destination CN -- )
Copies the parameters for all IE’s from one connection to another.

COPY_IE ( source CN\destination CN\IE identifier -- )
Copies parameters, relating to a specified IE, from one connection to another.

\NOTE
COPY_PARAMS and COPY_IE can be used to copy parameter values from the decoder instead of a connection by specifying #DECODER as the copy source. Using #DECODER specifies the IE’s of the last decoded layer 3 message.

Example:
Copy the parameters in the Bearer Capability IE of the last received message from the decoder into the current connection.

#DECODER ?CN I#BEARER_CAP COPY_IE

INIT_IE_BUFFERS ( -- )
Clears the contents of all IE buffers for the active connection. One buffer exists for each IE type. All IE buffers default to a 40 byte length.

LONG_BUFFER ( IE identifier -- )
Extends a buffer to 120 bytes. If an attempt is made to extend the buffers of more than three IE’s, an error message is displayed.

Example:
Assign a long buffer to the ‘Display’ IE for the active connection.
I#DISPLAY LONG_BUFFER

RESET_IE_MAP ( -- )
Resets all IE’s to the default (40 byte) length for the active connection.

RESET_ALL_IE_MAPS ( -- )
Resets all IE’s to the default length for all connections.

?CN_IE_BUFFER ( IE identifier -- address of buffer or 0 )
Returns the address of the buffer for the specified IE for the active connection. 0 is returned if the specified IE identifier is unknown.
COPY_BUFFERS ( source CN\dest CN -- )
Copies all buffers for generated IE’s from one connection to another.

COPY_IE_BUF ( source CN\dest CN\IE identifier -- )
Copies only a single IE buffer. If #DECODER is specified as the source, COPY_IE_BUF copies
the complete IE from the last decoded message (if it was present in that message) to the
destination connection.

Example:
Copy the Bearer Capability IE, if it was present in the last decoded layer 3 message, from the
decoder to connection 7.
#DECODER #CN7 I#BEARER_CAP COPY_IE_BUF

Connection Timer Management

Each connection in a layer 3 simulation can use four timers. The first timer is used exclusively for
the CCITT timer T312. The second timer is used for the CCITT defined timers. The third and
fourth timers are general purpose timers.

The first timer, CCITT timer T312, is started when the network side of the interface has sent a
broadcast SETUP message and is waiting for a valid response. This timer is specified by using
the identifier:

<table>
<thead>
<tr>
<th>Timer Identifier</th>
<th>Default Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>#T312</td>
<td>#T312_DUR – 4 seconds</td>
</tr>
</tbody>
</table>

The second timer is used for the other CCITT defined timers. The length of this timer is specified
by using one of the following identifiers:

<table>
<thead>
<tr>
<th>Timer Identifier</th>
<th>Default Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>#T302</td>
<td>#T302_DUR – 10 seconds</td>
</tr>
<tr>
<td>#T303</td>
<td>#T303_DUR – 4 seconds</td>
</tr>
<tr>
<td>#T304</td>
<td>#T304_DUR – 15 seconds</td>
</tr>
<tr>
<td>#T305</td>
<td>#T305_DUR – 30 seconds</td>
</tr>
<tr>
<td>#T306</td>
<td>#T306_DUR – 120 seconds</td>
</tr>
<tr>
<td>#T307</td>
<td>#T307_DUR – 180 seconds</td>
</tr>
<tr>
<td>#T308</td>
<td>#T308_DUR – 4 seconds</td>
</tr>
<tr>
<td>#T309</td>
<td>#T309_DUR – 90 seconds</td>
</tr>
<tr>
<td>#T310</td>
<td>#T310_DUR – 10 seconds</td>
</tr>
<tr>
<td>#T313</td>
<td>#T313_DUR – 4 seconds</td>
</tr>
<tr>
<td>#T318</td>
<td>#T318_DUR – 4 seconds</td>
</tr>
<tr>
<td>#T319</td>
<td>#T319_DUR – 4 seconds</td>
</tr>
</tbody>
</table>

Note that these timers refer to the same timer; only one of these timers can be used at once. The
general purpose timers can be used as required and are specified by the following identifiers:

<table>
<thead>
<tr>
<th>Timer Constant</th>
<th>Default Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>#USER1</td>
<td>#USER1_DUR – 10 seconds</td>
</tr>
<tr>
<td>#USER2</td>
<td>#USER2_DUR – 10 seconds</td>
</tr>
</tbody>
</table>
CN_DEFAULT_TIMERS ( -- )
   Resets all timer durations for the active connection to their default values.

INIT_DEFAULT_TIMERS ( -- )
   Resets all timer durations for all connections to their default values.

=TIMER_DUR ( timer identifier \ time -- )
   Sets the specified timer, in tenths of seconds, for all connections.

=CN_TIMER_DUR ( timer identifier \ time -- )
   Sets the timer duration, in tenths of seconds, for the specified timer for the active connection.

?CN_TIMER_DUR ( timer identifier -- time )
   Returns the timer duration, in tenths of seconds, associated with the given timer identifier.

CN_START_TIMER ( timer identifier -- )
   Starts the indicated timer.

CN_STOP_TIMER ( timer identifier -- )
   Stops the indicated timer.

?CN_TIMER ( timer identifier -- flag )
   Returns true when the timer associated with the given timer identifier has expired.

CN_STOP_ALL_TIMERS ( -- )
   Stops all four of the current connection timers.

?CN_LAST_TIMER ( -- timer identifier )
   Returns the timer identifier of the last timer that was started.

Example 1:
For the current connection, set the duration for timer T302 to 8 seconds and start the timer.
#T302 80 =CN_TIMER_DUR
#T302 CN_START_TIMER

Example 2:
Find out the duration of timer T302, and if it has expired.
#T302 ?CN_TIMER_DUR ( Leaves the duration on the stack )
#T302 ?CN_TIMER ( Leaves a flag on the stack indicating if the timer has expired )

Example 3:
For the current connection, start timer T312, T303, and the two user timers – one for 20 seconds, the other for 30 seconds.

#T312 CN_START_TIMER
#T303 CN_START_TIMER
#USER1 200 =CN_TIMER_DUR
#USER1 CN_START_TIMER
#USER2 300 =CN_TIMER_DUR
#USER2 CN_START_TIMER
18.3 B-Channel Management

The ISDN D-Channel protocol performs the signalling and negotiation for voice or data calls between the network and terminal. These calls are carried by a B-Channel which must be allocated and deallocated by the system as required.

The following commands do not connect or disconnect B-Channels to application processors. They simply provide a convenient mechanism for test scripts to manage the B-Channel.

BC_INIT ( -- )
Marks all B-Channels as free (unallocated).

BC_ALLOC ( B-Channel -- )
Flags the specified B-Channel as allocated.

BC_DISC ( B-Channel -- )
Flags the specified B-Channel as disconnected.

BC_FREE ( B-Channel -- )
Marks the B-Channel as free and disconnected.

BC_CONN ( B-Channel -- )
Marks the B-Channel as connected and allocated.

⚠ NOTE
The distinction between a channel being allocated and connected is that an allocated channel might not yet be connected, but it prevents that channel from being used by any other connection.

?BC_CONN ( B-Channel -- flag )
Returns true if the specified B-Channel is connected.

?BC_ALLOC ( B-Channel -- flag )
Returns true if the specified B-Channel is allocated.

?BC_SELECT ( -- B-Channel )
Returns the number of a B-Channel marked as free. If all B-Channels are in use, 0 is returned.

SET_INTERFACE ( interface -- )
Where: interface = #BASIC Basic Rate (2 B-Channels)
      #PRIMARY Primary Rate (30 B-Channels)

Sets the number of B-Channels, the call reference, length, and marks all channels free. For the Primary Rate interface, the call reference length is two octets and the number of B-Channels is 30, whereas for the Basic Rate interface, the call reference length is one octet and the number of B-Channels is 2.

⚠ NOTE
For Primary Rate, 30 channels are set even though North American T1 uses only 24.
18.4 Message Builder

ISDN layer 3 messages are complex. The message builder is used to create layer 3 messages. Such user-generated messages (whether valid or deliberately invalid) can be used to test the response of ISDN equipment.

Generating Messages

The programming method of message generation can dynamically build layer 3 messages. The message type and included information elements are specified when the message is created. Parameter field values, specific for the current connection, are used when each information element is built. Individual octets of an information element can be included or excluded when the IE is built. Thus, correct and incorrect messages can be generated and transmitted.

Messages built using the programming method or the interactive method (described in the basic User Manual) can be saved in a message pool for future retrieval and transmission. Messages built using the programming method can be transmitted immediately in a layer 2 I frame or UI frame.

There are two programming methods of message generation. In the first method, all IE's to be included in the message are specified.

**MESSAGE>** (message identifier -- )
Generates messages by specifying the message identifier and all desired information element identifiers. MESSAGE> is paired with <STORE to save the message into a pool buffer or with <SEND or <SEND UNIT to send the message in a layer 2 I frame or UI frame respectively. The IE identifiers are listed between MESSAGE> and <STORE, <SEND, or <SEND_UNIT. The pool buffer name string must be specified for <STORE.

**NOTE**
The information element buffer for each specified IE is used to build the message. Therefore, each element must be built before the message is built.

**NOTE**
The call reference value and call reference flag used when messages are built are sent with the =CN_CR_VALUE and =CN_CR_FLAG commands.

Example:
Build a setup message with the Bearer Capability, Channel Identification, and Called Party Number IE's and save the message into a buffer named SETUP#1.

```
M#SETUP MESSAGE>
  I#BEARER_CAP
  I#CHANNEL_ID
  I#CALLED_NUM
  "SETUP#1" <STORE
```
Example:
Send a status message with the cause and call state IE's in a UI frame.

M\#STATUS MESSAGE\>
I\#CAUSE
I\#CALL_STATE
<SEND_UNIT

The message builder can function in automatic mode (IE's are sorted into the correct order with Shift IE's as required), or manual mode (IE's of a message are included in a specified order).

AUTO_MODE ( -- )
Selects automatic mode (default) for message generation. Information elements are sorted in the correct order before the message is built. Any required Shift IE's are included in the message.

\[ NOTE \]
Locking Shift IE's are included if elements from codeset 5, 6, or 7 are included. Manual mode must be used to build the message if non-locking Shift IE's are desired.

MAN_MODE ( -- )
Selects manual mode for message generation. Information elements are built into the message in the order they appear between MESSAGE> and <STORE, <SEND, or <SEND_UNIT.

\[ NOTE \]
Shift IE's must be included if required. Temporary IE buffers can be used to build multiple occurrences of the same IE in a message.

An alternate method of message generation is to include all mandatory IE's and any selected optional IE's when the message is built. The message can then be sent as part of a layer 2 frame.

MESG ( message identifier -- address \ count )
Builds the indicated message, including all mandatory information elements and any previously selected optional or other information elements. The returned address and count are suitable for use with SEND_I or SEND_UI.

Send Data Source Menu
→ Data Source
L.3 Default Message function key
Modify Type function key

ALL_SELECT ( -- )
Includes all optional IE's in the message.

ALL_UNSELECT ( -- )
No optional IE's are included in the message.

SELECT ( IE identifier -- )
Includes the specified IE in the message.
UNSELECT (IE identifier -- )
The specified IE is not included in the message.

Example:
Send an alert message with all mandatory IE's and the channel identification IE in a layer 2 l frame.

ALL_UNSELECT
I#CHANNEL_ID SELECT
M#ALERT MESG SEND_I

Example:
Send a congestion control message with all mandatory and optional IE's in a layer 2 U1 frame.

ALL_SELECT
M#CON_CON MESG SEND_U1

\NOTE
The set of selected IE's is connection dependent.

Generating Information Elements

Information elements are generated in a similar way to which messages are generated. The included octets are specified when the IE is constructed. The parameter field values can be changed using meaningfully named identifiers.

Each connection has an associated data structure of parameter values that does not change unless specifically altered. Thus, it is possible to generate unique messages for each connection.

The parameter field data can be examined and modified using the *COD identifier in conjunction with the field selectors used for the current message set.

*COD ( -- address )
Returns the base address of the coder parameter field data structure for the current connection.

\NOTE
*COD is the message builder complement of the layer 3 decoder *DEC identifier.

Example:
Modify the Shift IE parameters.

#LOCKING *COD ->SH_TYPE !
#CODESET7 *COD ->SH_CODESET !
!STRING ( source string \ dest string -- )
Copies the source string to the destination string address.

Example:
Modify the display IE parameter.
" IDACOM calling" *COD ->D_DISPLAY !STRING

Once the parameter field values have been modified as applicable, the IE can be built.

ELEMENT> <ELEMENT ( IE identifier -- )
Builds the specified information element.

Octets are specified for inclusion or exclusion from the IE by using the following commands between ELEMENT> and <ELEMENT.

The information element is built into the buffer associated with the IE for the current connection. The ?CN_IE_BUFFER command can be used to access this buffer.

⚠️ NOTE
All desired information elements must be built with this command before they can be built into a message with the MESSAGE> or MESG commands.

ALL_INCLUDED ( -- )
Includes all applicable octets in the IE.

ALL_EXCLUDED ( -- )
No octets are included in the IE. The IE consists of only the IE identifier and IE length (0).

INCLUDED ( octet identifier -- )
Includes the specified octet when the IE is generated. The octet identifiers given for the ?L3_OCTET command are also used for this command.

EXCLUDED ( octet identifier -- )
The specified octet is not included when the IE is generated.

NUM_INCLUDED ( octet identifier \ n -- )
Includes the specified octet the given number of times. This command should only be used for octets that may be repeated.

Example:
Build a cause information element.

I#CAUSE ELEMENT>
   ALL_EXCLUDED
   OCTET_3 INCLUDED
   OCTET_4 INCLUDED
   OCTET_5 INCLUDED
<ELEMENT
Multiple occurrences of the same information element can be built. However, since only one IE can be stored in the current connection's buffer for each IE type, temporary buffers must be used to store other occurrences.

LOAD_IE_BUFFER ( temp buffer -- )
Saves the generated IE in the specified temporary buffer. LOAD_IE_BUFFER must be used between ELEMENT> and <ELEMENT. There are 8 temporary buffers: I#TEMP0 to I#TEMP7.

Example:
Build two keypad IE's. Store one in the I#TEMP0 buffer and the other in the I#KEYPAD buffer.

I#KEYPAD ELEMENT>
  ALL_INCLUDED
  X" 1" *COD ->K_KEYPAD !STRING
  I#TEMP0 LOAD_IE_BUFFER
<ELEMENT

I#KEYPAD ELEMENT>
  ALL_INCLUDED
  X" 2" *COD ->K_KEYPAD !STRING
<ELEMENT

Some information elements, such as undefined IE's, cannot be built with the message builder. It is still possible to create these IE's and save them in the current connection's buffers.

DEFINE_IE ( string \ IE identifier -- )
Stores the specified string in the IE buffer for the current connection. Usually, the temporary IE buffers are used with this command. The string must contain the entire contents of the IE including the IE code and length octets.

Example:
Build a user-defined information element.
X" 0103010203" I#TEMP0 DEFINE_IE

TEMP_CODESET ( codeset \ temp buffer -- )
Associates the IE in the specified temporary buffer with the given codeset. TEMP_CODESET must be used when building messages with temporary buffers in automatic mode. The IE's are sorted by codeset in automatic mode.

Example:
Send an INFO message with the user-defined information element in temporary buffer 0.
AUTO_MODE
7 I#TEMP0 TEMP_CODESET
M/INFO MESSAGE>
  I#TEMP0
<SEND
The following example illustrates how to build three information elements and a message using these IE's.

Build a bearer capability IE with:
- CCITT coding standard;
- speech transfer capability;
- circuit transfer mode;
- 64 kbit/s transfer rate;
- default structure;
- point-to-point configuration;
- demand establishment; and
- bidirectional symmetry.

I#BEARER_CAP ELEMENT> ( Build Bearer Cap with octets 3, 4, 4a & 4b )

( Specify IE structure )

ALL_EXCLUDED ( Exclude all possible octets )
OCTET_3 INCLUDED ( Octet 3 )
OCTET_4 INCLUDED ( Octet 4 )
OCTET_4A INCLUDED ( Octet 4A )
OCTET_4B INCLUDED ( Octet 4B )

( Modify IE field data )

#CCITT *COD -> BC_CODING_STANDARD ! ( Octet 3 )
#SPEECH *COD -> BC_TRANSFER_CAP !
#CIRCUIT_MODE *COD -> BC_TRANSFER_MODE ! ( Octet 4 )
#64KBIT/S *COD -> BC_TRANSFER_RATE !
#DEFAULT *COD -> BC_STRUCTURE ! ( Octet 4a )
#POINT_TO_POINT *COD -> BC_CONFIGURATION !
#DEMAND *COD -> BC_ESTABLISHMENT !
#BIDIRECT_SYMMETRIC *COD -> BC_SYMMETRY ! ( Octet 4b)
#64KBITS/S *COD -> BCTRANSFER_RATE_4B !
<ELEMENT ( Generates IE into Bearer Cap default IE buffer )

Build a channel identification IE using the existing values, but replace the channel map with the channel map in the last received message.

I#CHANNEL_ID ELEMENT> ( Build Channel Id IE with only new channel map )

( Use previously specified IE structure )

*DEC -> CID_MAP ( Address of channel map from received message )
*COD -> CID_MAP ! STRING ( Store in coder variable )
<ELEMENT ( Generate IE into Channel Id Buffer )
Build a display information element IE using the existing values, but change the display string information and store it into a temporary buffer.

I#DISPLAY ELEMENT> ( Generate IE into a temporary buffer )

( Use previously specified IE structure )
" Test string for Display IE" ( Enter string to be stored )
*COD ->D_DISPLAY !STRING ( Modify Display information )

I#TEMPO LOAD_IE_BUFFER ( Specify temporary buffer 0 for IE generation )
0 I#TEMPO TEMP_CODESET ( Specify codeset for temporary buffer 0 )
<ELEMENT ( Generate Display IE into temporary buffer 0 )

Build a setup message using the IE's built in the previous three examples.

AUTO_MODE
M#SETUP MESSAGE>
  I#CHANNEL_ID
  I#BEARER_CAP
  I#TEMPO
  " SETUP" <STORE

Connection Independent Management Message Transmission

This type of message is normally generated by the layer 3 management procedure. Therefore, a management message must be sent independently of the connection.

SM_MSG ( message identifier -- )
Transmits a layer 3 management message. The values used to build the message are independent of each connection. The message identifier is the identifier used in the current message set to refer to the desired management message type. Only the following message types should be used for management message types:

RESTART
  call reference value: Global (0)
  call reference flag: Origination (0)
  channel identifier IE: only if restart class is 0 or 6
  restart indicator IE

RESTART ACKNOWLEDGE
  call reference value: Global (0)
  call reference flag: Destination (1)
  channel identifier IE: only if restart class is 0 or 6
  restart indicator IE

RELEASE
  call reference value: taken from last received message
  call reference flag: inverse of flag in last received message
  cause IE

RELEASE COMPLETE
  call reference value: taken from last received message
  call reference flag: inverse of flag in last received message
  cause IE
RESUME REJECT
- Call reference value: taken from last received message
- Call reference flag: inverse of flag in last received message
- Cause IE

CONGESTION CONTROL
- Call reference value: taken from last received message
- Call reference flag: inverse of flag in last received message
- Congestion level IE
- Cause IE

STATUS
- Call reference value: taken from last received message
- Call reference flag: inverse of flag in last received message
- Cause IE
- Call state IE

The values used for the information elements sent in these management messages can be set with the following commands. Q.931 standard encoding is used for these information elements.

SYS_Q931_CAUSE (class \ value -- )
- Sets the cause class (octet 4, bits 5 through 7) and cause value (octet 4, bits 1 through 4) for the management Cause IE. Other fields are coded as follows:
  - Octets 3 and 4 are included.
  - Coding Standard: octet 3, bits 6, 7
  - Location: octet 3, bits 1–4
  - 0 (CCITT standardized coding)
  - 2 (public network serving the local user)
  - Network side
  - 0 (user) – user side

\NOTE
Some message sets combine the cause class and cause value into a single cause value field. The symbolic constants for the cause values cannot be used in these message sets.

SYS_Q931_REST_IND (restart class -- )
- Sets the restart class field (octet 3, bits 1 through 3) for the management Restart Indicator IE.
SYS_Q931_CHAN_ID (channel number -- )
Sets the channel number for the management Channel Identification IE. This value corresponds to the information channel selection field (octet 3, bits 1 through 2) for Basic Rate or the channel number field (octet 3.3, bits 1 through 7) for Primary Rate. Other fields are coded as follows:

Basic Rate:
Only octet 3 is included.
Int. id. present octet 3, bit 7 0 (implicitly identified)
Int. type octet 3, bit 6 0 (basic interface)
Pref./Excl. octet 3, bit 4 1 (exclusive)
D–channel ind. octet 3, bit 3 0 (not the D–channel)

Primary Rate:
Octets 3, 3.2, and 3.3 are included.
Int. id. present octet 3, bit 7 0 (implicitly identified)
Int. type octet 3, bit 6 1 (other interface)
Pref./Excl. octet 3, bit 4 0 (preferred)
D–channel ind. octet 3, bit 3 0 (not the D–channel)
Info. chan. sel. octet 3, bits 1–2 1 (as indicated)
Coding standard octet 3.2, bits 6–7 0 (CCITT standardized coding)
Number/Map octet 3.2, bit 5 0 (number)
Channel type octet 3.2, bits 1–4 3 (B–channel units)

SYS_Q931_CONG_LVL (congestion level -- )
Sets the congestion level field (octet 1, bits 1 through 4) for the management Congestion Level IE.

SYS_Q931_CALL_ST (call state -- )
Sets the call state field (octet 3, bits 1 through 6) for the management Call State IE. The coding standard field is set to 0 (CCITT standardized coding).

Example:
Send a layer 3 management Congestion Control message using the CCITT message set.

#RECEIVER_NOT_READY SYS_Q931_CONG_LVL
2 10 SYS_Q931_CAUSE
M#CON_CON SM_MSG
18.5 Message Buffer Pool Management

The message buffer pool management manages a collection of layer 3 messages. An array of entries, each of which contains a name field (used to identify the message), a comment field, and a data field (containing the message itself), is called a message pool. Both the name and comment fields are stored in the pool entry as normal strings, while the data field uses the first two bytes to denote the message length. The maximum number of buffers in the message buffer pool is 60.

The length of a message that can be stored in a single buffer is 130 octets. However, an extension mechanism exists whereby two adjacent buffers can be used to store an extended message (up to 262 bytes). The operation of this mechanism is fully automatic, and occurs whenever a message is stored which is too large to fit in a single buffer. An empty buffer or a message contained in one or two buffers is called a pool entry.

The user interface to the message buffer pool management enables access to specific, pre-defined layer 3 messages in test manager scenarios. These pre-defined messages are made with the message builder and are stored to disk using the message buffer pool management.

Specifically, the message buffer pool provides commands to address the name, comment, and data fields of any entry in the message pool. It is possible to initialize single pool entries or the entire pool. Individual message strings can be loaded or recalled to the pool and the entire message pool can be stored or recalled from disk files.

`?MESSAGE ( string -- address or 0 )`
Returns the data field address of the first message pool entry that matches the specified name, or 0 if no match is found.

`?MESSAGE_NUMBER ( string -- n | -1 )`
Returns the number of the first message pool entry that matches the specified name, or -1 if no match is found.

`=MESSAGE_NAME ( string\n -- )`
Assigns a name to the specified message pool entry.

Example:
Assign the name 'SETUP#5' to message pool entry 2.
```
" SETUP#5" 2 =MESSAGE_NAME
```

`?MESSAGE_NAME ( n -- address of n\textsuperscript{th} name field or 0 )`
Returns the name field address of the specified message pool entry, or 0 if an invalid entry number is specified.

\textbf{NOTE}
Message pool entries are numbered 0 through 59 although Pool Menu entries are numbered 1 through 60.
=MESSAGE_COMMENT ( string\n -- )  
Assigns a comment to the specified message pool entry.

Example:
  "Setup message with invalid Bearer Capability IE" 3 =MESSAGE_COMMENT

?MESSAGE_COMMENT ( n -- address of nth comment field or 0 )  
Returns the comment field address of the specified message pool entry, or 0 if an invalid entry number is specified.

?MESSAGE_DATA ( n -- address of nth data field or 0 )  
Returns the data field address of the specified message pool entry, or 0 if an invalid entry number is specified.

?MESSAGE_FREE ( n -- flag )  
Returns true if the specified message pool entry is currently unused (not occupied by either a complete or part of an extended message), or false if an invalid entry number is specified.

STORE_MESSAGE ( address\length\n -- )  
Stores a message starting at 'address' and of 'length' bytes in the data field of the specified message pool entry. If the message is too large to fit in one buffer entry, the buffer is automatically extended to use the next entry. If the next entry is unavailable, an error message will be displayed and no part of the message will be stored.

Example:
following the receipt of an I frame with a layer 3 message (as determined from within a test script), save the data contents of the frame for later use as pool entry 3.

L3-POINTER @ L3-LENGTH @
3 STORE_MESSAGE

RECALL_MESSAGE ( address\n -- length or -1 )  
Copies the data field of the specified message pool entry to the specified address and returns the length (in bytes) of the message, or -1 if an invalid entry number is specified.

CLEAR_MESSAGE ( n -- )  
Clears the name, comment, and data fields of the specified message pool entry. If the message was automatically extended, the next entry will also be cleared.

SAVE_MESSAGES ( filename -- )  
Saves the entire message buffer pool to the specified file.

Example:
Save the message buffer pool into the file 'POOL#2'.
  "DR0:POOL#2" SAVE_MESSAGES

LOAD_MESSAGES ( filename -- )  
Loads the message buffer pool from the specified file after checking the file size and type.
CLEAR_POOL ( -- )
Clears the name, comment, and data fields of all message pool entries.

WARNING
Use CLEAR_POOL with caution.

ALLOC_MESSAGE ( flag -- n or -1 )
For a flag value of 0, returns the number of the first unused pool entry, or -1 if no such entry can be found in the pool. A flag value of 1 indicating that a double length entry is required (for extended messages), returns the first of two successive unused pool entries, or -1 if none can be found.

COMPACT_POOL ( -- n )
Moves all used message pool entries to the bottom of the pool (i.e. closer to entry 0) and returns the number of used message pool entries.

CONCAT_POOLS ( filename -- )
Compacts the existing message pool and concatenates the pool contained in the specified file to the existing one. If the two pools are too large to fit in one pool, the pool in the specified file will be truncated, and an error message is displayed.

INSERT_ENTRY ( n -- )
Inserts a blank pool entry before the specified message pool entry. The specified subsequent message pool entries are shuffled upwards in the pool. If the last entry in the pool is in use, or an invalid entry is specified, the pool will remain unaltered and an error message will be displayed.

COPY_MESSAGE ( string
-- )
Copies the entire contents of the pool entry with the specified name into the specified message pool entry.

Example:
Copy the contents of the pool entry named ‘ALERT#1’ into message pool entry 4.
"ALERT#1" 4 COPY_MESSAGE

MOVE_MESSAGE ( name string
-- )
Copies the entire contents of the pool entry with the specified name into the specified message pool entry and deletes the source entry.

=POOL_CR_VALUE ( string
\call ref value -- )
Assigns a call reference value to the pool entry with the specified name. If the dummy call reference is in use for the specified message, the entry will remain unchanged and an error message will be displayed. The call reference value will be suitably truncated if the total number of octets available for the value is insufficient.

Example:
"CONNECT-MSG" 23 =POOL_CR_VALUE

?POOL_CR_VALUE ( string
-- call ref value )
Returns the call reference value used by the pool entry with the specified name. If the dummy call reference is in use for the specified message, an error message will be displayed.
=POOL_CR_FLAG ( string\call ref flag -- )
   Where: call ref flag = #DEST - destination
          #ORIG - origination

Sets the call reference flag for the pool entry with the specified name. If the dummy call reference is in use for the specified message, the entry will remain unchanged and an error message will be displayed.

Example:
Set the call reference flag to destination for the pool entry named 'RELEASE#2'.
   " RELEASE#2" #DEST =POOL_CR_FLAG

?POOL_CR_FLAG ( string -- call ref flag or -1 )
   Where: call ref flag = #DEST - destination
          #ORIG - origination

Returns the call reference flag used by the pool entry with the specified name. If the dummy call reference is in use for the specified message, an error message will be displayed.

ALTER_CALL_REF ( string\call ref. flag\call ref. value -- )
   Where: call ref flag = #DEST - destination
          #ORIG - origination

Changes only the call reference value and flag of the pool entry with the specified name. If the dummy call reference is in use for the specified message, an error message will be displayed.

Example:
   " SETUP#1" #ORIG 44 ALTER_CALL_REF

POOL ( string -- address \ length )
Returns the address and length of the pool entry with the specified name.

Example 1:
Send a buffer named 'Alert' in an information frame.
   " Alert" POOL SEND_I

Example 2:
Send a buffer named 'Alert' inside an unnumbered information frame.
   " Alert" POOL SEND_UI

Send Data Source Menu
   → Data Source
      Message Pool function key
          → Pool Entry Name
The following example illustrates how to build a message containing information elements, and use the message pool.

Example:
Build an ALERT message with the following information elements:

Channel Identification IE
- Interface identifier: implicitly identified
- Interface type: Basic Rate
- Preferred/exclusive: exclusive, only the indicated channel is acceptable
- D–Channel indicator: the channel identified is not the D–Channel
- Information channel selection: B1–Channel

Called Party Number IE
- Number type: national number
- Numbering plan: ISDN numbering plan Rec. E.164
- Number: 555–1212

I#CHANNEL_ID ELEMENT>
   ALL EXCLUDED
   OCTET_3 INCLUDED
   #IMPLICIT *COD ->CID_INT_PRESENT !
   #BASIC_INTERFACE *COD ->CID_INT_TYPE !
   #EXCLUSIVE *COD ->CID_PREF/EXCL !
   #NOT_D_CHANNEL *COD ->CID_DCHANNEL !
   #B1_CHANNEL *COD ->CID_INFO_CHAN_SEL !
</ELEMENT

I#CALLED_NUM ELEMENT>
   ALL EXCLUDED
   OCTET_3 INCLUDED
   OCTET_4 INCLUDED
   #NATIONAL *COD ->CLDN_NUMBER_TYPE !
   #ISDN_PLAN *COD ->CLDN_NUMBERING_PLAN !
   " 5551212" *COD ->CLDN_NUMBER !STRING
</ELEMENT

ISDN Programmer’s Manual
The X.25 PLP (Packet Layer Procedure) Emulation operates as a layer 3 emulation connected to the ISDN Basic Rate D-Channel layer 2. The following features are supported to interface to the layer 2 emulation:

- One SAPI is assigned to X.25 packet mode operation. The default value for this SAPI is 16.
- 255 simultaneous logical channel connections.
- Each logical channel can be connected to any of the 8 link connections.

### 19.1 Emulation Configuration

#### Setup Menu

<table>
<thead>
<tr>
<th>Emulation Mode</th>
<th>Protocol Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCE</td>
<td>STD=NONE</td>
</tr>
</tbody>
</table>

**Figure 19-1 Setup Menu**

- **Emulation Mode**
  - **DTE_END** (---)
    - Selects a logical DTE emulation mode. This is the default when the Basic Rate configuration is set as user.

  ![DTE function key](image)

- **DCE_END** (---)
  - Selects a logical DCE emulation mode. This is the default when the Basic Rate configuration is set to network.

  ![DCE function key](image)

- **Protocol Standard**
  - Selects a protocol standard for emulation.

  **STD=NONE** (---)
  - Conforms to a combination of the CCITT X.25 (1980/1984) Recommendation. The behaviour can be modified by the user.

  ![NONE function key](image)
STD=X25(80) (~ -- ~)
Conforms to the CCITT X.25 (1980) Recommendation and sets up all the appropriate system parameters.

X.25(1980) function key

STD=X25(84) (~ -- ~)
Conforms to the CCITT X.25 (1984) Recommendation (default) and sets up all the appropriate system parameters.

X.25(1984) function key

NOTE
The protocol standard selection affects some emulation parameters and procedures. See Table 19-1 for a list of affected parameters/procedures.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Link Procedure</td>
<td>Single Link or Multi-link</td>
<td>Single Link</td>
<td>Single Link or Multi-link</td>
</tr>
<tr>
<td>Sequence Number</td>
<td>Modulo 8 or 128</td>
<td>Modulo 8</td>
<td>Modulo 8 or 128</td>
</tr>
<tr>
<td>Poll Bit for SABM, SABME, and DISC</td>
<td>P=0 or P=1</td>
<td>P=0</td>
<td>P=0 or P=1</td>
</tr>
<tr>
<td>Unsolicited DM</td>
<td>Supported or Not Supported</td>
<td>Not Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Maximum User Data in DATA Packet</td>
<td>4096 Octets</td>
<td>1024 Octets</td>
<td>4096 Octets</td>
</tr>
<tr>
<td>Facility Field Length</td>
<td>109 Octets</td>
<td>63 Octets</td>
<td>109 Octets</td>
</tr>
<tr>
<td>Call User Data</td>
<td>16 Octets</td>
<td>16 Octets</td>
<td>16 Octets</td>
</tr>
<tr>
<td></td>
<td>128 Octets with fast select facility</td>
<td>128 Octets with fast select facility</td>
<td></td>
</tr>
<tr>
<td>Call Connected Format</td>
<td>Basic or Extended</td>
<td>Basic</td>
<td>Basic or Extended</td>
</tr>
<tr>
<td>Clear Request Format</td>
<td>Basic or Extended</td>
<td>Basic</td>
<td>Basic or Extended</td>
</tr>
<tr>
<td>Clear Confirmation Format</td>
<td>Basic or Extended</td>
<td>Basic</td>
<td>Basic or Extended</td>
</tr>
<tr>
<td>Maximum Interrupt Data</td>
<td>32 Octets</td>
<td>1 Octet</td>
<td>32 Octets</td>
</tr>
</tbody>
</table>

Table 19-1 Protocol Standard Parameters/Procedures

Packet Layer
IDACOM's X.25 Emulation implements an automatic layer 3 state machine (refer to Section 19.3 for more information). Depending on the emulation mode selected, either the DCE or DTE Packet Layer Menu is displayed. DTE emulation uses timers T20 to T23; DCE emulation uses timers T10 to T13. All other configuration commands are used by both emulation modes.
Packet Layer:

- Emulation: AUTOMATIC
  - T20 Timer (Sec): 180.0
- Max Data Size: 128
- Sequence Numbering: MOD 128
  - T21 Timer (Sec): 200.0
  - T22 Timer (Sec): 180.0
  - T23 Timer (Sec): 180.0

Figure 19–2 DTE Packet Layer Menu

- **Emulation**
  - L3_ON (---)
    - Activates the layer 3 state machine (default) resulting in automatic responses to received packets.
    - **AUTOMATIC** function key

- **L3_OFF (---)**
  - Deactivates the layer 3 state machine resulting in no automatic responses to received packets.
    - **MANUAL** function key

- **Max Data Size**
  - L3_MAX (---)
    - Specifies the maximum number of bytes in the data field of transmitted or received data packets for all logical channels. Valid values are 0 through 4100 (default is 128).

  Example:
  - Set the maximum data field size to 256.
  - `256 = SIZE`

- **NOTE**
  - The maximum frame size should be sufficiently larger than the maximum data size to allow for the address and control fields plus the data packet header.

- **Sequence Numbering**
  - Packet_MOD8 (---)
    - Selects modulo 8 method of sequence numbering for the packet layer on all LCN's.
    - **MOD 8** function key

  Packet_MOD128 (---)
    - Selects modulo 128 method of sequence numbering (default) for the packet layer on all LCN's.
    - **MOD 128** function key
**NOTE**

When the monitor detects a GFI (general format identifier) of either modulo 8 or 128, the corresponding decoding and reporting mechanism is used.

When the emulation detects a GFI that does not match that set by these commands:
- a DTE emulation ignores the received packet; and
- a DCE emulation responds with a diagnostic packet indicating the reception of an invalid GFI.

The following timers are used for DTE emulation.

→ **T20 Timer (Sec)**

**T20-VALUE ( address )**

Contains the duration, in tenths of seconds, of the T20 timer (default is 180 seconds). The T20 timer is started when the emulation is a DTE and a restart request is transmitted using T20-VALUE to set timeout. This timer is not used when the value is set to 0. For appropriate action by the DTE when this timer expires, see TABLE D-2/X.25 in the CCITT X.25 (1984) Recommendation.

Example:
Set the T20 timer to 240 seconds.

2400 T20-VALUE !

Modify T20 Restart Timer function key

→ **T21 Timer (Sec)**

**T21-VALUE ( address )**

Contains the duration, in tenths of seconds, of the T21 timer (default is 200 seconds). The T21 timer is started when the emulation is a DTE and a call request is transmitted using T21-VALUE to set timeout. This timer is not used when the value is set to 0. For appropriate action by the DTE when this timer expires, see TABLE D-2/X.25 in the CCITT X.25 (1984) Recommendation.

Example:
Set the T21 timer to 240 seconds.

2400 T21-VALUE !

Modify T21 Call Timer function key

→ **T22 Timer (Sec)**

**T22-VALUE ( address )**

Contains the duration, in tenths of seconds, of the T22 timer (default is 180 seconds). The T22 timer is started when the emulation is a DTE and a reset request is transmitted using T22-VALUE. This timer is not used when the value is set to 0. For appropriate action by the DTE when this timer expires, see TABLE D-2/X.25 in the CCITT X.25 (1984) Recommendation.

Example:
Set the T22 timer to 1 minute.

600 T22-VALUE !

Modify T22 Reset Timer function key
\( \rightarrow T23 \text{ Timer (Sec)} \)

**T23-VALUE** ( -- address)

Contains the duration, in tenths of seconds, of the T23 timer (default is 180 seconds). The T23 timer is started when the emulation is a DTE and a clear request is transmitted using T23-VALUE. This timer is not used when the value is set to 0. For appropriate action by the DTE when this timer expires, see TABLE D-2/X.25 in the CCITT X.25 (1984) Recommendation.

Example:
Set T23 timer to 4 minutes.
2400 T23-VALUE!

Modify T23 Clear Timer function key

The following timers are used for DCE emulation.

<table>
<thead>
<tr>
<th>Packet Layer:</th>
<th>AUTOMATIC</th>
<th>T10 Timer (Sec)</th>
<th>60.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emulation</td>
<td></td>
<td>T11 Timer (Sec)</td>
<td>180.0</td>
</tr>
<tr>
<td>Max Data Size</td>
<td>128</td>
<td>T12 Timer (Sec)</td>
<td>60.0</td>
</tr>
<tr>
<td>Sequence Numbering</td>
<td>MOD 128</td>
<td>T13 Timer (Sec)</td>
<td>60.0</td>
</tr>
</tbody>
</table>

Figure 19-3 DCE Packet Layer Menu

\( \rightarrow T10 \text{ Timer (Sec)} \)

**T10-VALUE** ( -- address)

Contains the duration, in tenths of seconds, of the T10 timer (default is 60 seconds). The T10 timer is started when the emulation is a DCE and a restart indication is transmitted using T10-VALUE to set timeout. This timer is not used when the value is set to 0. For appropriate action by the DCE when this timer expires, see TABLE D-1/X.25 in the CCITT X.25 (1984) Recommendation.

Example:
Set the T10 timer to 2 minutes.
1200 T10-VALUE!

Modify T10 Restart Timer function key
→ **T11 Timer (Sec)**

**T11-VALUE** ( -- address )

Contains the duration, in tenths of seconds, of the T11 timer (default is 180 seconds). The T11 timer is started when the emulation is a DCE and an incoming call is transmitted using T11-VALUE to set the timeout. This timer is not used when the value is set to 0. For appropriate action by the DCE when this timer expires, see TABLE D-1/X.25 in the CCITT X.25 (1984) Recommendation.

Example:
Set the T11 timer to 200 seconds.
2000 T11-VALUE !

Modify T11 Call Timer function key

→ **T12 Timer (Sec)**

**T12-VALUE** ( -- address )

Contains the duration, in tenths of seconds, of the T12 timer (default is 60 seconds). The T12 timer is started when the emulation is a DCE and a reset indication is transmitted using T12-VALUE to set timeout. This timer is not used when the value is set to 0. For appropriate action by the DCE when this timer expires, see TABLE D-1/X.25 in the CCITT X.25 (1984) Recommendation.

Example:
Set the T12 timer to 30 seconds.
300 T12-VALUE !

Modify T12 Reset Timer function key

→ **T13 Timer (Sec)**

**T13-VALUE** ( -- address )

Contains the duration, in tenths of seconds, of the T13 timer (default is 60 seconds). The T13 timer is started when the emulation is a DCE and a clear indication is transmitted using T13-VALUE to set timeout. This timer is not used when the value is set to 0. For appropriate action by the DCE when this timer expires, see TABLE D-1/X.25 in the CCITT X.25 (1984) Recommendation.

Example:
Set the T13 timer to 90 seconds.
900 T13-VALUE !
## Facilities

<table>
<thead>
<tr>
<th>Facility Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>➔ Call Request Facility</td>
</tr>
<tr>
<td>User Defined Facility</td>
</tr>
<tr>
<td>Call Accept/Connect</td>
</tr>
<tr>
<td>Call Accept Facility</td>
</tr>
<tr>
<td>User Defined Facility</td>
</tr>
<tr>
<td>Call User Data</td>
</tr>
<tr>
<td>Clear User Data</td>
</tr>
</tbody>
</table>

**Figure 19-4 Facility Menu**

**SET_FAC_LEN** (n -- )

Specifies the maximum facility length of the received packet. Valid values are 0 through 63 for the 1980 Recommendation, and 0 through 109 for the 1984 Recommendation.

**➔ Call Request Facility**

**NO_FAC** (--- )

No facility data is included on any of the 255 channels when transmitting call request/incoming call packets (default).

[![NONE function key]](image)

**YES_FAC** (--- )

Automatically negotiates data packet size, packet window, throughput class, and fast select on all 255 channels when transmitting call request/incoming call packets.

[![NEGOTIATE function key]](image)

**USER_FAC** (--- )

Enables facility negotiation on all 255 channels. Facility fields defined with the MAKE_FAC command are transmitted within call request/incoming call packets.

[![USER DEFINED function key]](image)
**CLASS** (throughput class -- )

Specifies the throughput facility class on all 255 channels. This facility is used in a call request/incoming call facility negotiation. It expects a numerical value as an input parameter and has no output parameters. Valid values are 75, 150, 300, 600, 1200, 2400, 4800, 9600 (default), 19200, and 48000.

Example:
Set the throughput facility class to 2400.
2400 =CLASS

**User Defined Facility**

**MAKE_FAC** (string -- )

Specifies the facility field used in transmitted call request/incoming call packets when USER_FAC is called. The first byte in the string is the facility length, with following bytes used as facilities. These facilities should be defined in hex byte format as shown in the example. The maximum length of the facility field is 109 octets.

Example:
Define a facility for a packet size negotiation of 256.
X"03420808" MAKE_FAC

The first byte in the hex string (03) indicates that the following facility field has a length of 3 bytes.

The second byte (42) indicates that the facility being negotiated is the packet size.

Bytes 3 and 4 (0808) indicate that the packet size is 256 for both the called DTE and calling DTE.

Modify Facility function key

**Call Accept/Connect**

**YES_CA** (---)

Uses the address field (default), received in call request/incoming call packets, as the address field in transmitted call accept/connect packets on all 255 channels.

USE ADDRESS function key

**NO_CA** (---)

No address, facility, or call user data fields are used in transmitted call accept/connect packets.

NO ADDRESS function key
Call Accept Facility

NO_CAFAC (---)
The facility field in call accept/connect packets is not used.

NONE function key

ECHO_CAFAC (---)
Echoes the facility field (default) received in call request/incoming call packets in the call accept/connect packet on all 255 channels.

ECHO function key

USER_CAFAC (---)
Enables facility negotiation on all 255 channels. Facility fields defined with the MAKE_CAFAC command are transmitted within call accept/connect packets.

USER_DEFINED function key

User Defined Facility

MAKE_CAFAC (string ---)
Specifies the facility field used in transmitted call accept/connect packets when USER_CAFAC is called. The first byte in the string is the facility length, with following bytes to be used as facilities. These facilities should be defined in hex byte format as shown in the example. The maximum length of the facility field is 63 octets for the 1980 Recommendation, and 109 octets for the 1984 Recommendation.

Example:
Define a facility for a packet size negotiation of 256 in call accept packets.
X"03420808" MAKE_CAFAC

The first byte in the hex string (03) indicates that the following facility field has a length of three bytes.

The second byte (42) indicates that the facility being negotiated is packet size.

Bytes 3 and 4 (0808) indicate that the packet size is 256 for both the called and calling DTE.

Modify Facility function key
→ **Call User Data**

**MAKE_CUD** (string -- )

Specifies the call user data field used in transmitted call request/incoming call and call accept/connect packets on all 255 channels (maximum length is 64 bytes). This field is also used in clear request packets when extended format is used.

Example 1:
Define a call user data field that contains 11 characters.
```
X"C000000003010025800064" MAKE_CUD
```

Example 2:
Clear the call user data field.
```
NO MAKE_CUD
```

Modify Call User Data function key

→ **Clear User Data**

**MAKE_CLRUD** (string -- )

Specifies the clear user data field used in transmitted clear request indication packets on all 255 channels (maximum length is 64 bytes).

Example 1:
Define a clear user data field that contains 11 characters.
```
X"C000000003010025800064" MAKE_CLRUD
```

Example 2:
Clear the clear user data field.
```
NO MAKE_CLRUD
```

Modify Clear User Data function key
LCN Setup

The X.25 Emulation supports 255 logical channels which can be set to any of 4096 LCN's (logical channel numbers). Logical channels are assigned and configured for link, SVC/PVC, called address, calling address, packet window size, and data echo from LCN Setup Menu 1.

<table>
<thead>
<tr>
<th>LCN</th>
<th>LINK</th>
<th>TYPE</th>
<th>Called Address</th>
<th>Calling Address</th>
<th>Window</th>
<th>Echo</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH1</td>
<td>1</td>
<td>SVC</td>
<td>43042001</td>
<td>33001001</td>
<td>2</td>
<td>OFF</td>
</tr>
<tr>
<td>CH2</td>
<td>2</td>
<td>SVC</td>
<td>43042001</td>
<td>33001001</td>
<td>2</td>
<td>OFF</td>
</tr>
<tr>
<td>CH3</td>
<td>3</td>
<td>SVC</td>
<td>43042001</td>
<td>33001001</td>
<td>2</td>
<td>OFF</td>
</tr>
<tr>
<td>CH4</td>
<td>4</td>
<td>SVC</td>
<td>43042001</td>
<td>33001001</td>
<td>2</td>
<td>OFF</td>
</tr>
<tr>
<td>CH5</td>
<td>5</td>
<td>SVC</td>
<td>43042001</td>
<td>33001001</td>
<td>2</td>
<td>OFF</td>
</tr>
<tr>
<td>CH6</td>
<td>6</td>
<td>SVC</td>
<td>43042001</td>
<td>33001001</td>
<td>2</td>
<td>OFF</td>
</tr>
<tr>
<td>CH7</td>
<td>7</td>
<td>SVC</td>
<td>43042001</td>
<td>33001001</td>
<td>2</td>
<td>OFF</td>
</tr>
<tr>
<td>CH8</td>
<td>8</td>
<td>SVC</td>
<td>43042001</td>
<td>33001001</td>
<td>2</td>
<td>OFF</td>
</tr>
<tr>
<td>CH9</td>
<td>9</td>
<td>SVC</td>
<td>43042001</td>
<td>33001001</td>
<td>2</td>
<td>OFF</td>
</tr>
<tr>
<td>CH10</td>
<td>10</td>
<td>SVC</td>
<td>43042001</td>
<td>33001001</td>
<td>2</td>
<td>OFF</td>
</tr>
</tbody>
</table>

Figure 19-5 LCN Setup Menu 1

CH (logical channel -- )
   Specifies the logical channel to use as the current channel. Valid values are 1 through 255.

CH1 (---)
   Uses logical channel 1 as the current channel. The CH2 to CH255 commands function in the same manner for selecting the 255 available channels.

The LCN of the currently selected logical channel can also be changed using the =LCN command.

⚠️ NOTE
   If more than one channel has the same LCN value, the automatic emulation uses the first one found on the LCN Setup Menus. Errors in protocol can be forced by sending out packets on a second channel with the same LCN by using the CH1 to CH255 commands.

=LCN (LCN --)
   Specifies the LCN of the currently selected logical channel. Valid values are 0 through 4095.

Example:
   Set the LCN on channel 30 to 225.
   CH30 225 =LCN

Modify LCN function key
=LCNCES ( link -- )
Sets the link CES (connection endpoint suffix) for the currently selected channel.

Example:
Set the link CES on channel 30 to 1.
CH30 1 =LCNCES

Modify LINK function key

=ALL_LCNCES ( link -- )
Sets the link CES (connection endpoint suffix) for all 255 channels.

LCNCES ( -- address )
Contains the link CES (connection endpoint suffix) for the currently selected channel. This value can be stored with the ! (store) or read with the @ (fetch) operations.

Example:
CH1 LCNCES @

In this case, the value of the link CES for channel 1 is left on the stack.

SVC ( -- )
Sets the currently selected channel as an SVC (switched virtual circuit).

Example:
Set channel 3 as SVC.
CH3 SVC

SVC function key

PVC ( -- )
Sets the currently selected channel as a PVC (permanent virtual circuit).

Example:
Set channel 2 as PVC.
CH2 PVC

PVC function key

The called and calling addresses can be changed using the =CALLED and =CALLING commands. The corresponding address fields can be cleared using the LCNCALLED and LCNCALLING strings.

=CALLED (" string" -- )
Specifies the called address field for subsequent call request packets on the currently selected logical channel. Up to 15 decimal digits can be specified within the ASCII string.

Example:
Define a called address of 1234567890 on channel 2.
CH2 "1234567890" =CALLED

Modify Called function key
LCNCALLED ( -- address )
Contains a 16 byte string identifying the called address field defined with the =CALLED command. The first byte of this string contains the length of the called digits.

In the previous example, where the called address is defined as 1234567890, LCNCALLED contains the following hex values:

0A31323334353637383930

The first byte is the length of the defined called address (hex 0A, decimal 10). The remaining bytes are the hex values for ASCII representation of decimal 1234567890.

Example:
Clear the called address field on channel 3 (by setting the length to 0).
CH3 0 LCNCALLED !

Modify Called function key

=CALLING ( " string" -- )
Specifies the calling address field for subsequent call request packets on the currently selected logical channel. Up to 15 decimal digits can be specified within the ASCII string.

Example:
Define a calling address of 1234567890 on channel 2.
CH2 " 1234567890" =CALLING

Modify Calling function key

LCNCALLING ( -- address )
Contains a 16 byte string identifying the calling address field defined with the =CALLING command. In the previous example, where the calling address is defined as 1234567890, LCNCALLING contains the following hex values: 0A31323334353637383930.

The first byte of the length is the defined calling address (hex 0A, decimal 10). The remaining bytes are the hex values for ASCII representation of decimal 1234567890.

Example:
Clear the calling address field on channel 3.
CH3 0 LCNCALLING !

Modify Calling function key

=WINDOW ( window size -- )
Specifies the maximum packet window size of the currently selected logical channel. Valid values are 1 through 7 for modulo 8, and 1 through 127 for modulo 128 (default is 2).

Example:
Set the maximum packet window size to 7 for logical channel 10.
CH10 7 =WINDOW

Modify Window function key
ECHO_ON (--) 
Echoes the received data packet on the currently selected channel.

Example:
Echo all data packets on CH55.
CH55 ECHO_ON

ECHO_OFF (--) 
Disables the data packet echo (default) on the currently selected channel.

Example:
Turn off the data packet echo enabled in the previous example.
CH55 ECHO_OFF

Each of the 255 logical channels can be configured for fast select facility, clear request format, and clear confirm format from the LCN Setup Menu 2.

<table>
<thead>
<tr>
<th>LCN</th>
<th>Fast Select</th>
<th>Clear Request</th>
<th>Clear Confirm</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH1 1</td>
<td>OFF</td>
<td>Not Extended</td>
<td>Not Extended</td>
</tr>
<tr>
<td>CH2 2</td>
<td>OFF</td>
<td>Not Extended</td>
<td>Not Extended</td>
</tr>
<tr>
<td>CH3 3</td>
<td>OFF</td>
<td>Not Extended</td>
<td>Not Extended</td>
</tr>
<tr>
<td>CH4 4</td>
<td>OFF</td>
<td>Not Extended</td>
<td>Not Extended</td>
</tr>
<tr>
<td>CH5 5</td>
<td>OFF</td>
<td>Not Extended</td>
<td>Not Extended</td>
</tr>
<tr>
<td>CH6 6</td>
<td>OFF</td>
<td>Not Extended</td>
<td>Not Extended</td>
</tr>
<tr>
<td>CH7 7</td>
<td>OFF</td>
<td>Not Extended</td>
<td>Not Extended</td>
</tr>
<tr>
<td>CH8 8</td>
<td>OFF</td>
<td>Not Extended</td>
<td>Not Extended</td>
</tr>
<tr>
<td>CH9 9</td>
<td>OFF</td>
<td>Not Extended</td>
<td>Not Extended</td>
</tr>
<tr>
<td>CH10 10</td>
<td>OFF</td>
<td>Not Extended</td>
<td>Not Extended</td>
</tr>
</tbody>
</table>

Figure 19-6 LCN Setup Menu 2

The following commands define the use and format of facility fields.

FAST_SELECT_OFF (--) 
Disables the fast select facility (default) on the currently selected logical channel.

Example:
Turn off the fast select facility on channel 2.
CH2 FAST_SELECT_OFF

ISDN Programmer’s Manual
FAST_SELECT_ON (---)
Enables the fast select facility with no restrictions in call request packets on the currently selected logical channel.

Example:
Turn on the fast select facility for channel 2.
CH2 FAST_SELECT_ON

ON function key

FAST_SELECT_RESTRICTION (---)
Enables the fast select facility with restriction on the currently selected logical channel. When the FAST_SELECT_RESTRICTION is active and a call request packet is received, the emulation transmits a clear request packet which contains address, facility, and user data fields.

Example:
Activate the fast select facility for channel 3.
CH3 FAST_SELECT_RESTRICTION

WITH RESTRICTION function key

Clear request packets contain calling and called address fields, facility fields, and optional clear user data in fast select mode if the CLEARREQ_EXT command is issued.

CLEARREQ_NOT_EXT (---)
Extended format in the transmitted clear request packets is not used on the currently selected logical channel. This is the default mode for all logical channels and applies to both DCE and DTE ends.

Example:
Turn off extended format in clear request packets on channel 1 for transmission.
CH1 CLEARREQ_NOT_EXT

NOT EXTENDED function key
CLEARREQ_EXT (--)  
Uses extended format in the transmitted clear request packets on the currently selected logical channel. When this command is issued, the diagnostic code field, address length fields, and facility length fields must be present in the clear request packet. The clear user data field is optional.

This command is selected per channel and applies to both DTE and DCE ends. It remains in effect until CLEARREQ_NOT_EXT is issued.

Example:
Use extended format in clear request packets on channel 4.
CH4  CLEARREQ_EXT

CLEAR REQUEST EXTENDED function key

NOTE
If the emulation partner is also an IDACOM tester, this command should also be issued on the partner for the same logical channel.

CLEARCONF_NOT_EXT (--)  
Extended format in the transmitted clear confirmation packets is not used on the currently selected logical channel. This is the default mode for all logical channels and applies to the DCE end.

Example:
Turn off extended format in clear confirmation packets on channel 1.
CH1  CLEARCONF_NOT_EXT

NOT EXTENDED function key

CLEARCONF_EXT (--)  
Uses extended format in the transmitted clear confirmation packets on the currently selected logical channel. When this command is issued, the diagnostic code field, the address length fields, and the facility length fields must be present in the clear confirm packet.

The DTE side can receive, but not transmit, a clear confirm packet in extended format. The DCE side can receive or send clear confirm packets in extended format. This command remains in effect until CLEARCONF_NOT_EXT is issued.

Example:
Use extended format in clear confirmation packets on channel 4.
CH4  CLEARCONF_EXT

CLEAR CONFIRM EXTENDED function key

NOTE
If the emulation partner is also an IDACOM tester, this command should also be issued on the partner for the same logical channel.
19.2 Emulation Decode

This section describes the data flow diagram for the emulation decode and lists the variables in which decoded information is saved.


![Diagram of X.25 Emulation Data Flow Diagram - Decode]

**Figure 19-7 X.25 Emulation Data Flow Diagram – Decode**

**Communication Variables**

This section describes both receive and transmit communication variables. Receive variables are set during the decode process, and contain protocol specific information as defined in the X.25 Recommendation. The emulation uses the information in these variables to determine the appropriate action to external events.

Transmit variables are used when transmitting a frame and can be used in test scripts to modify the emulation response.

▲ **NOTE**

*These variables can be read using the @ (fetch) operation.*

**PR** ( -- address )

Contains the P(R) (packet receive sequence number) of the last received data, RR, RNR, or REJ packet. Valid values are 0 through 7 for modulo 8, and 0 through 127 for modulo 128. Bits 6 to 8 of the packet identifier octet in data, RR, RNR, or REJ packets represent modulo 8. Bits 2 to 8 of octet 4 represent modulo 128.

**PS** ( -- address )

Contains the P(S) (packet send sequence number) of the last received data packet. Valid values are 0 through 7 for modulo 8, and 0 through 127 for modulo 128. Bits 6 to 8 of the packet identifier octet in data packets for modulo 8. Bits 2 to 8 of octet 3 in data packets for modulo 128.
RCAUSE ( -- address )
Contains the received cause byte. Octet 4 of the following packets:
- Clear request/indication
- Reset request/indication
- Restart request/indication

If the emulation is a DCE, the X.25 Emulation checks if the cause byte has a value of 0, or is between 128 and 255. See the CCITT X.25 (1984) Recommendation for defined values for specific packets.

RDIAG ( -- address )
Contains the diagnostic byte of the received packet. Valid values are 0 through 255. Octet 5 of the following packets:
- Clear request/indication
- Reset request/indication
- Restart request/indication

\[ \text{\textbullet\ NOT} \]
See the CCITT X.25 (1980/1984) Recommendations for defined values.

REC_PKT_ID ( -- address )
Contains the received packet identifier byte. Used by the emulation to determine if this is an appropriate packet to receive for current LCN state. See Table B-2 for valid values.

RECD ( -- address )
Contains the D (delivery confirmation) bit of a received data packet, call request/incoming call, or call accept/connect packet (0 or 1). Bit 7 of the first packet octet.

If this bit is set to 1 in any other packet type, the decode operation indicates an invalid GFI.
If the emulation is a DCE, it responds with a diagnostic packet with a value of 40 as the diagnostic code indicating an invalid GFI.

RECM ( -- address )
Contains the M (more) bit of a received data packet (0 or 1). Bit 5 of octet 3 of the data packet represents modulo 8, and bit 1 of octet 4 represents modulo 128. When the M bit is set to 1, more data will follow.

RECPKTMOD ( -- address )
Contains the received packet modulo. Bits 5 and 6 of the first octet of the packet.
RECPKTMOD contains the hex value 10 for modulo 8, and 20 for modulo 128. Any other value is decoded as an invalid GFI. If the emulation is a DCE, it responds with a diagnostic packet with a value of 40 as the diagnostic code indicating an invalid GFI.

RECQ ( -- address )
Contains the Q (qualifier) bit of a received data packet (0 or 1). Bit 8 of the first data packet octet.

If this bit is set to 1 in any other packet type, the decode operation indicates an invalid GFI.
If the emulation is a DCE, it responds with a diagnostic packet with a value of 40 as the diagnostic code indicating an invalid GFI.
RIUD (--- address)
Contains the user data byte in a received interrupt packet. Octet 4 in interrupt packets. The
interrupt user data field can be 1 to 32 octets. If the emulation is in state p1 – Ready, p2 –
Calling, or p5 – Collision, the emulation responds with a clear request packet. If the
emulation is in state d1 – Data or d8 – Remote Busy and the interrupt user data field is
greater than 32 octets in length, the emulation responds with a reset request packet.

RLCN (--- address)
Contains the combined logical group identifier and logical channel number of the received
packet. Valid values are 0 through 4095. Bits 1 to 4 of the first packet octet and all bits of
the second packet octet.

The emulation supports up to 255 logical channels at a time. These are defined on the LCN
Setup Menu 1 as CH1 through CH255. Any changes to this should be made prior to running
the emulation.

DCE Emulation If RLCN contains a value other than that defined on the LCN Setup Menu 1,
the emulation ignores this received packet. If in a state waiting restart confirm or link up, it
responds with a diagnostic packet.

DTE Emulation If RLCN contains a value other than that defined on the LCN Setup Menu 1,
the emulation ignores this received packet regardless of the state.

SCAUSE (--- address)
Contains the cause field for transmitted clear request/indication, reset request/indication, and
restart request/indication packets. Used in the automatic emulation and the S:CLEARR,
S:RESETR, and S:RESTARTR commands. When the automatic emulation detects an error,
SCAUSE contains the following values:

<table>
<thead>
<tr>
<th>Packet Type</th>
<th>DCE Emulation Local Procedure Error</th>
<th>DTE Emulation DTE Originated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>Reset</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Restart</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

SDIAG (--- address)
Contains the diagnostic field for transmitted clear request/indication, reset request/indication,
and restart request/indication packets. Used in the automatic emulation and the S:CLEARR,
Recommendation, Annex E for defined values. After the packet is transmitted, SDIAG is reset
to 0.
SDIAG-EXP ( -- address )
Contains the diagnostic explanation field used in the S:DIAG command. The first byte
contains the length of this field (up to 3 bytes). The remaining bytes contain the first 3 or less
octets of the last received packet. To change this field in a transmitted packet, change the
value in this variable just prior to transmission.

Example 1:
If a clear request packet is received on LCN 64, SDIAG-EXP contains the hex value:

```
03 10 40 13
```

Example 2:
Indicate that a short packet (2 bytes) was received on LCN 1.

```
0X02100100 SDIAG-EXP !
```

SENDM ( -- address )
Contains the value of the M bit sent in the next data packet (default is 0.)

SIUD ( -- address )
Contains the interrupt user data byte used when sending interrupt request packets. Octet 4 in
interrupt request packets (default is 0). Maximum values are 32 octets for CCITT X.25 (1984); 1
octet for CCITT X.25 (1980).

STATE_L3 ( -- address )
Contains a value which represents one of the layer 3 states in the layer emulation (see the
'Emulation State Machines' on page 19–25). Valid values are 1 through 4.

SET_INT_LEN (# -- )
Sets the maximum length of interrupt user data of the received frame/packet. Maximum
length is 32 octets for 1984, and octet for 1980.

SENDQB ( -- address )
Contains the Q (qualifier) bit for transmitted packets (0 or 1). Bit 8 of the first packet octet. If
this bit is set to 1 in a packet type other than a data packet during DTE emulation, the
partner should respond with a diagnostic packet with a diagnostic code of 40, indicating an
invalid GFI.

SENDDB ( -- address )
Contains the D (delivery confirmation) bit of a transmitted packet (0 – default or 1). Bit 7 of
the first packet octet. The emulation automatically sets it to 1 if a call request/incoming call
packet is received with a D bit of 1. SENDDB is reset to 0 when the call accept/connect
packet is transmitted. See RECD for expected response from the emulation partner.
SENDGFI ( -- address )
Contains the GFI modulo indicator used for transmitting packets. Bits 5 and 6 of the first packet octet. The default value is hex 10 for packet modulo 8, and hex 20 for packet modulo 128.

RCALLED ( -- address )
Contains a 16 byte string identifying the called address field of a received packet. It is used by the S:CALLR and S:CALLC commands. See the example under RCALLING; string contents and handling are similar.

RCALLING ( -- address )
Contains a 16 byte string identifying the calling address field of a received packet. It is used by the S:CALLR and S:CALLC commands.

Example:
Check whether the CALLING number of the last received call request matches a pre-defined number (eg. 43042001). The ?MATCH command is used to determine if the calling address in a received call request packet matches the defined address.

TCLR

#IFDEF MATCH-CALL
0 VARIABLE MATCH-CALL 12 ALLOT
(MATCH-CALL will contain the desired calling address)
#ENDIF

0 STATE_INIT{
"43042001"
COUNT (Obtain # of digits in calling address)
15 MIN (Set maximum to 15 digits)
DUP MATCH-CALL C! (Write # of digits to first byte of matching string)
MATCH-CALL 1 + SWAP CMOVE (Write calling address in matching string)
}STATE_INIT

0 STATE{
R*CALLREQ 1 ?RX
ACTION{
RCALLING (Get this calling address)
COUNT (Get # of digits in this calling address)
MATCH-CALL (Get desired matching address)
?MATCH (Compare addresses)
IF
BEEP (Notify user)
T."Address Match has occurred." TCR
ENDIF
}ACTION
}STATE
RCUD (-- address)
Contains a 128 byte string identifying the call user data of a received packet.

Example:
The tester receives a call request packet with a call user field that contains 11 characters, i.e. the hex characters C00000003010025800064.

Obtain the length of the call user data field (in this case 11).
RCUD C@

If the call user data field is present, its use and format are determined by bits 7 and 8 of the first octet. This octet can be obtained by RCUD 1 + C@. Refer to CCITT 1984 Recommendation X.244 for further information on call user data.

RFAC (-- address)
Contains a 256 byte string identifying the facility field of a received call request/incoming call, call accept/connect, clear request/clear indication, or clear confirmation packet.

\[\text{\textbf{NOTE}}\]
The maximum facility length is 109 octets. If the received length is longer than 109 octets, the X.25 Emulation transmits a clear request packet.

Example:
The tester receives a call request packet with a facility field that contains 8 characters, i.e. the hex characters 02AA420808430303.

Obtain the length of the facility field (8 in this case).
RFAC C@

Obtain the first octet of the first facility (hex 02 in this case). This indicates that the first facility is:
- a Class A facility, i.e. it is followed by a two octet parameter field; and
- throughput class negotiation.
RFAC 1 + C@

Obtain the throughput class octet (hex AA in this case). This indicates that the throughput class for the called DTE and calling DTE is 9600 bits.
RFAC 2 + C@

Obtain the first octet of the second facility (hex 42 in this case). This indicates that the second facility is:
- a Class B facility, i.e. it is followed by a two octet parameter field; and
- packet size.
RFAC 3 + C@

Obtain the packet size for the called DTE (hex 08 in this case). This indicates a packet size of 256.
RFAC 4 + C@

Obtain the packet size for the calling DTE (hex 08 in this case). This indicates a packet size of 256.
RFAC 5 + C@
Obtain the first octet of the third facility (hex 43 in this case). This indicates that the third facility is:
- a Class B facility, i.e. it is followed by a two octet parameter field; and
- window size.

RFAC 6 + C@

Obtain the window size of the called DTE (hex 03 in this case). This indicates a window size of 3.

RFAC 7 + C@

Obtain the window size of the calling DTE (hex 03 in this case). This indicates a window size of 3.

RFAC 8 + C@

**NOTE**
Refer to the CCITT X.25 (1984) Recommendation, Section 7 Formats for Facility Fields and Registration Fields for further information on decoding facilities.

**SET_FAC_LENGTH** ( -- )
Sets the maximum length of the facility field in received call/clear packets.

The X.25 layer 3 Emulation program supports simultaneous execution of 255 logical channels.

The logical channel commands access parameters from the currently selected logical channel (default is CH1). The current logical channel can be set using the following methods:
- Under the X25_Send topic, press the Enter LCN function key. If the entered value (0 through 4095) matches one of those defined on LCN Setup Menu 1, the value in the LCN variable will be set to that entered value.
- Use one of the CH1 to CH255 commands (see the 'LCN Setup' section on page 19-11).
- Use the =LCN command (see the 'LCN Setup' section on page 19-11).

**LCN** ( -- address )
Contains a pointer to the value of the current logical channel and logical group. Valid values are 0 through 4095 (default is 1).

**LCNCES** ( -- address )
Contains the value of link CES for the current channel (default is 0). Valid values are 0 through 7.

**LCNSTATE** ( -- address )
Contains a pointer to the current LCN state with one of the following values:
- 1 – p1 Ready, 2 – p2 Calling, 3 – p5 Collision, 4 – d1 Data, 5 – d2 Reset, 6 – p6 Clearing,
- 8 – d8 Remote Busy (see the 'Emulation State Machines' section on page 19-25).

**LCNWINDOW** ( -- address )
Contains a pointer to the window size for the current logical channel. Valid values are 1 through 7 for modulo 8, and 1 through 127 for modulo 128 (default is 2).

**LCNSIZE** ( -- address )
Contains a pointer to the size of the data field used with the DATA command. Valid values are 0 through 4110 bytes (default is 128).
LCNSIZE ( -- address )
Contains a pointer to the maximum size of the data field in data packets for the current logical channel. This is negotiated in the facility field during call setup. The standard negotiated value is 128 octets; others are 16, 32, 64, 256, 512, 1024, 2048, and 4096 octets.

NVS ( -- address )
Contains a pointer to the current LCN P(S) (send sequence number) used when a data packet is transmitted. It is set to 0 when the logical channel has just entered the flow control ready state – d1 and incremented by one every time a data packet is transmitted. Valid values are 0 through 7 for modulo 8, and 0 through 127 for modulo 128.

NVR ( -- address )
Contains a pointer to the current LCN P(R) (receive sequence number). This is the next P(S) expected to be received. It is set to 0 when the logical channel has just entered the flow control ready state – d1. When a data packet is received with a valid P(S) and P(R), NVR is incremented by one. Valid values are 0 through 7 for modulo 8, and 0 through 127 for modulo 128. NVR is used when transmitting data, RR, RNR, and REJ packets.

NSU ( -- address )
Contains a pointer to the LCN P(S) (sequence number) of the lowest unacknowledged data packet. It is set to 0 when the logical channel has just entered the flow control ready state – d1. When a data packet is received with a valid P(S) and P(R) or an RR, RNR, or REJ packet is received with a valid P(R), the current received P(R) is written in NSU. Valid values are 0 through 7 for modulo 8, and 0 through 127 for modulo 128.

19.3 Emulation Response
The X.25 layer 3 Emulation is implemented as a multi-layer, state-driven protocol emulation. There are separate program modules for each protocol layer. These modules communicate with each other to implement protocol response behavior.

The emulation has been set up to run as:
- an automatic simulation which operates precisely in accordance with the CCITT X.25 (1980/1984) Recommendations;
- a semi-automatic tester. The test manager is used to build and execute test scenarios to test responses and for generation of errors (see Section 20); and
- a manual tester. The test is controlled from the user's keyboard.
Emulation State Machines

To ensure correct protocol operation, state machines have been implemented. Based on input events (i.e. received frames or packets), transitions from one state to another are made in accordance with CCITT Recommendations.

**NEW_L3_STATE** (n --)
Sets the layer 3 state machine to a specific state. Valid values are 1 through 4. The state value is stored in the STATE_L3 variable.

Example:
Put the layer 3 state machine in the idle state.

```
1 NEW_L3_STATE
```

<table>
<thead>
<tr>
<th>State Number</th>
<th>State Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Idle</td>
<td>Disconnected from layer 2</td>
</tr>
<tr>
<td>2</td>
<td>Waiting Operational</td>
<td>Waiting for link operational</td>
</tr>
<tr>
<td>3</td>
<td>Waiting Restart Confirm</td>
<td>Waiting for restart confirm</td>
</tr>
<tr>
<td>4</td>
<td>Link Up</td>
<td>Operational state</td>
</tr>
</tbody>
</table>

Table 19–2  Layer 3 States

Embedded in state 4 of the layer 3 state machine, is a state machine for each logical channel.

**NEW_LCN_STATE** (n --)
Sets the logical channel state of the currently selected LCN to a specific state. Valid values are 1, 2, 3, 4, 5, 6, and 8.

Example:
Put the logical channel state machine in state p1 – Ready.

```
1 NEW_LCN_STATE
```

**NOTE**
The layer 3 state machine should be in State 4 (i.e. link up) when using the NEW_LCN_STATE command.
<table>
<thead>
<tr>
<th>State Number</th>
<th>State Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>p1 - Ready</td>
<td>SVC LCN is ready to send or receive call</td>
</tr>
<tr>
<td>2</td>
<td>p2 - Calling</td>
<td>SVC LCN has sent call request, waiting for call connect</td>
</tr>
<tr>
<td>3</td>
<td>p5 - Collision</td>
<td>DTE and DCE simultaneously transmit a call request and incoming call on the same logical channel</td>
</tr>
<tr>
<td>4</td>
<td>d1 - Flow Control Ready</td>
<td>Flow control state: Ready for data</td>
</tr>
<tr>
<td>5</td>
<td>d2 - Reset</td>
<td>LCN has sent reset request, waiting for reset confirm</td>
</tr>
<tr>
<td>6</td>
<td>p6 - Clearing</td>
<td>SVC LCN has sent clear request, waiting for clear confirm</td>
</tr>
<tr>
<td>8</td>
<td>d8 - Remote Busy</td>
<td>Flow control ready with remote end busy</td>
</tr>
</tbody>
</table>

**Table 19-3 Logical Channel States**

**Automatic Responses**

The state machines normally handle the protocol automatically, i.e. there are automatic responses to received frames and packets.

The following commands activate and deactivate the frame and packet state machines.

**X25_L3_OFF (--)**
Deactivates the X.25 layer 3 state machine. No automatic responses to received packets are generated. This command is useful for layer 3+ simulation.

**X25_L3_ON (--)**
Activates the X.25 layer 3 state machine. Automatic responses to received packets are generated. This command is useful for layer 4+ simulation.

**EMUL_ON (--)**
Activates all state machines (default). Automatic responses to received frames and packets are generated.

**EMUL_OFF (--)**
Deactivates all state machines. No automatic responses to received frames and packets are generated.
State-Dependent Send Commands

Either function keys or commands are used to transmit frames or packets in conjunction with the automatic state machines. These commands or function keys force protocol state changes and the emulation thereby expects the correct response.

**NOTE**

When using these commands or function keys, the layer 2 and layer 3 protocol state machines must be activated. See Section 21 for examples.

The currently selected logical channel is:
- the last channel activated by the last CHn command;
- the value entered with the Enter LCN function key under the X.25_Send topic; or
- the last logical channel specific packet received.

**CALL (---)**

If the layer 3 and LCN state machines are in a state that allows a call packet to be sent, the emulation transmits a call request packet for a DTE emulation, or an incoming call packet for a DCE emulation. This packet is sent out on the currently selected LCN with the called and calling addresses as specified on the LCN Setup Menu.

The T21/T11 timer is started and the LCN state machine is set to state p2 - Calling. In response, the emulation expects a call connect packet if configured as DTE and a call accept if configured as DCE. If a correctly formatted call connect/accept packet is received, the LCN state machine goes to state d1 - Data and the T21/T11 is stopped. If the call connect/accept packet has an error in the facility field, a clear request packet is transmitted.

Example:
```
CALL CH2 CALL
```

(Send call request/incoming call on current LCN)

(Sends call request/incoming call on channel 2)

**CLEAR (---)**

If the layer 3 or LCN state machines are in a state that allows a clear packet to be sent, the emulation transmits a clear request packet for a DTE emulation, or a clear indication packet for a DCE emulation. This packet is sent out on the currently selected LCN.

The T23/T13 timer is started and the LCN state machine is set to state p6 - Clearing. In response, the emulation expects a clear confirm packet. If a correctly formatted clear confirm packet is received, the LCN state machine goes to state p1 - Ready and the T23/T13 is stopped.

Example:
```
CLEAR CH3 CLEAR
```

(Sends Clear request/indication on currently selected LCN)

(Sends Clear request/indication on channel 3)
DATA ( -- )
If the layer 3 or LCN state machines are in a state that allows a data packet to be sent and the packet layer window is open, the emulation transmits a data packet with the correct P(S) and P(R) with the maximum amount of data as specified on the Packet Layer Menu. This packet is sent out on the currently selected LCN. For an SVC (switched virtual circuit), a call must have been set up first.

The data field is filled with pre-defined text, and can be changed using the DEFINE_DATA command. A test program can determine if the window is open by using the WINDOW? command.

Example:
```
DATA       ( Sends Data packet on the currently selected LCN )
CH4 DATA   ( Sends Data packet on channel 4 )
```

DEFINE_DATA ( filename -- )
Defines the text within the data field of the data packet for the DATA command and the DATA function key. A file containing the desired text must first be created using the editor on the Home processor. DEFINE_DATA overwrites BUFFER 0 of the test manager (see the 'Using Buffers' section on page 20–16).

\* NOTE
If the file created by the user contains less than 1024 characters, the data field is padded out to 1024 characters.

Example:
Create a file with the name CUSTOM.F with the desired text.

- Press the HOME key.
- Move the topic bar to the Files topic.
- Insert a formatted floppy diskette in drive 0.
- Press the Edit function key.

The 'Edit script:' prompt is displayed.

- Type: DR0:CUSTOM.F.
- Press \( \Leftarrow \) (RETURN).
- Enter desired text.
- Press the Save function key.
- Press the Quit function key.

Follow the instructions in the User Manual for loading the X.25 Emulation. Switch to the application processor which is running the program.

To use the previously created file:

- Press the ESC key to enter the command mode.
- Type: DR0 " CUSTOM.F" DEFINE_DATA
- Press the ESC key to leave the command mode.
Any data packets sent using the DATA function key under the X.25_Send topic will contain the text created by the user.

**INTERRUPT ( -- )**
If the layer 3 or LCN state machines are in a state that allows an interrupt packet to be sent, the emulation transmits an interrupt packet. This packet is sent out on the currently selected LCN provided a call has been set up.

Example:

```
INTERRUPT ( Sends interrupt packet on currently selected LCN )
CH5  INTERRUPT ( Sends interrupt packet on channel 5 )
```

**RESET ( -- )**
If the layer 3 or LCN state machines are in a state that allows a reset request packet to be sent, the emulation transmits a reset request packet for a DTE emulation or a reset indication packet for a DCE emulation. This packet is sent out on the currently selected LCN.

The T22/T12 timer is started and the LCN state machine is set to state d2 – Reset. In response, the emulation expects a reset confirm packet. If a correctly formatted reset confirm packet is received, the LCN state machine goes to state d1 – Data and T22/T12 is stopped.

Example:

```
RESET ( Sends reset packet on currently selected LCN )
CH6  RESET ( Sends reset packet on channel 6 )
```

**RESTART ( -- )**
If the layer 3 state machine is in a state that allows a restart request packet to be sent, the emulation transmits a restart request packet for a DTE emulation or a restart indication packet for a DCE emulation. This packet is sent out on logical channel 0.

The T20/T10 timer is started and the layer 3 state machine is set to the waiting restart confirm state. In response, the emulation expects a restart confirm packet. If a correctly formatted restart confirm packet is received, the layer 3 state machine goes to the Link Up state and the LCN state machine goes to state p1 – Ready for an SVC or state d1 – Data for a PVC. T20/T10 is stopped and CH1 is selected as the current LCN.

**NOTE**

RESTART, CALL, DATA, RESET, and CLEAR packets can also be transmitted by pressing the corresponding function key under the X.25_Send topic.

The following command transmits user–defined data (80 characters maximum).

**SENDD ( string -- )**

Transmits a user–defined data field in a data packet. The defined string is transmitted in a data packet containing the correct P(S) and P(R).

SENDD only transmits a data packet if the corresponding layer 3 window is open and sufficient acknowledgements have been received. The status of the layer 3 window is determined using the WINDOW? command.

If the layer 3 window is not open, the data is discarded as there is no queuing implemented at this level.
Example:
X"10010B22300000000000" SENDP  ( Transmits a call request packet on LCN 1 )
"A quick brown fox jumped over the lazy dog" SENDD
( Transmits a data packet on current LCN with this text in data field )

State-Independent Send Commands

S:RESTARTR  (-- )
Transmits a restart request packet on logical channel 0 using values from the following variables:
LCN  - Logical group and channel value (0)
SENDQB - Q bit value
SENDDB - D bit value
SENDGFI - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
SCAUSE - Cause value
SDIAG - Diagnostic value

S:RESTARTC  (-- )
Transmits a restart confirm packet on logical channel 0 using values from the following variables:
LCN  - Logical group and channel value (0)
SENDQB - Q bit value
SENDDB - D bit value
SENDGFI - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)

S:CALLR  (-- )
Transmits a call request/incoming call packet on the selected channel using values from the following variables:
LCN  - Logical group and channel value
SENDQB - Q bit value
SENDDB - D bit value
SENDGFI - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
LCNCALLED - Called address field with the first byte containing the length of address and followed by the address bytes in ASCII
LCNCALLING - Calling address field with the first byte containing the length of address and followed by the address bytes in ASCII
FACILITY-ACTIVE - Indication of type of facility negotiation desired
0 = Facilities not wanted
1 = Negotiate facilities for size, window, and throughput class
2 = Send facilities defined with MAKE_FAC command.
CUD-BUFFER - Call user data field buffer. The first byte is the length of the call user data field followed by the call user data
->CID_INFO_CHAN_SEL
  #NO_CHANNEL
  #AS_INDICATED
  #B1_CHANNEL
  #B2_CHANNEL
  #ANY_CHANNEL
->CID_INT_ID
  ( hex characters )
->CID_CODING_STANDARD
  #CCITT
  #INTERNATIONAL
  #NATIONAL
  #NETWORK_SPECIFIC
->CID_NUMBER/MAP
  #NUMBER
  #MAP
->CID_CHANNEL/MAP_TYPE
  #B_CHANNEL_UNITS
  #H0_CHANNEL_UNITS
  #H11_CHANNEL_UNITS
  #H12_CHANNEL_UNITS
->CID_MAP
  ( hex characters )
->CID_NUMBER
  ( numeric value )

Info. chan. sel., Octet 3
  no channel
  as indicated
  B1 channel
  B2 channel
  any channel
Interface ident., Octet 3.1 *
  max. length 8 octets
Coding standard, Octet 3.2
  CCITT
  other international standards
  national standard
  standard defined for the network
Number/Map, Octet 3.2
  number
  map
Chan./Map type, Octet 3.2
  B-channel units
  H0-channel units
  H11-channel units
  H12-channel units
Slot map, Octet 3.3 *
  max. length 4 octets
Channel number, Octet 3.3
  range 0 through 127

Congestion Level IE (I#CONG_LEVEL)

->CL_CONGESTION_LEVEL
  #RECEIVER_READY
  #RECEIVER_NOT_READY
Congestion level
  receiver ready
  receiver not ready

Date/Time IE (I#DATE/TIME)

Possible octet inclusions/exclusions:

OCTET_3, OCTET_4, OCTET_5, OCTET_6, OCTET_7, OCTET_8

->DT_YEAR
  ( numeric value )
  Year, Octet 3
  range 0 through 255
->DTMONTH
  ( numeric value )
  Month, Octet 4
  range 0 through 255
->DTDAY
  ( numeric value )
  Day, Octet 5
  range 0 through 255
->DTHOUR
  ( numeric value )
  Hour, Octet 6
  range 0 through 255
play IE (I#DISPLAY)

Decimal octet inclusions/exclusions:

T.3
.DISPLAY
IA5 characters

max. length 80 octets

-to-End Transit Delay IE (I#END-END_DELAY)

Decimal octet inclusions/exclusions:

T.3, OCTET.3A, OCTET.3B, OCTET.4, OCTET.4A, OCTET.4B, OCTET.5, OCTET.5A, T.5B

.CUMUL_DELAY
.REQ_DELAY
.MAX_DELAY

Cum. transit delay, Octet 3, 3a, & 3b
Req. end-end delay, Octet 4, 4a, & 4b
Max. end-end delay, Octet 5, 5a, & 5b

Point Identifier IE (I#ENDPOINT_ID)

Decimal octet inclusions/exclusions:

T.3, OCTET.4

.USID
.INTERPRETER
.MATCHES_USID+TID
.MATCHES_USID
.TID

User service id., Octet 3
Interpreter, Octet 4
matches USID and TID
matches USID and not TID
Terminal identifier, Octet 4

range 0 through 127
range 0 through 63
S:CALLC (---)
Transmits a call connected/accepted packet on selected channel using values from the following variables:

- **LCN** - Logical group and channel value
- **SENDQB** - Q bit value
- **SENDDB** - D bit value
- **SENDGFI** - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
- **RCALLED** - Called address field with the first byte containing the length of address and followed by the address bytes in ASCII
- **RCALLING** - Calling address field with the first byte containing the length of address and followed by the address bytes in ASCII
- **CAFAC-ACTIVE** - Indication of the type of facility to use:
  - 0 = Do not send facilities
  - 1 = Echo received facilities from buffer RFAC
  - 2 = Send facilities defined by MAKE_CAFAC command

S:CLEARC (---)
Transmits a clear request/indication packet on the selected channel using values from the following variables:

- **LCN** - Logical group and channel value
- **SENDQB** - Q bit value
- **SENDDB** - D bit value
- **SENDGFI** - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
- **SCAUSE** - Cause value
- **SDIAG** - Diagnostic value

When the fast select facility is active, the following are also included in clear request packets.

- **LCNCALLED** - Called address field
- **LCNCALLING** - Calling address field
- **FACILITY-ACTIVE** - Indication of type of facility negotiation desired (see S:CALLR)
- **CUD-BUFFER** - Call user data field buffer. The first byte is the length of the call user data field followed by the call user data

S:CLEARC (---)
Transmits a clear confirmation packet on the selected channel using values from the following variables:

- **LCN** - Logical group and channel value
- **SENDQB** - Q bit value
- **SENDDB** - D bit value
- **SENDGFI** - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)

S:RESETR (---)
Transmits a reset request/indication packet on the selected channel using values from the following variables:

- **LCN** - Logical group and channel value
- **SENDQB** - Q bit value
- **SENDDB** - D bit value
- **SENDGFI** - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
- **SCAUSE** - Cause value
- **SDIAG** - Diagnostic value
S:RESETC (--)  
Transmits a reset confirmation packet on the selected channel using values from the following variables:
LCN - Logical group and channel value
SENDQB - Q bit value
SENDDB - D bit value
SENDGFI - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)

S:INTR (--)  
Transmits an interrupt packet on the selected channel using values from the following variables:
LCN - Logical group and channel value
SENDQB - Q bit value
SENDDB - D bit value
SENDGFI - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
SIUD - Interrupt user data byte

S:INTC (--)  
Transmits an interrupt confirmation packet on the selected channel using values from the following variables:
LCN - Logical group and channel value
SENDQB - Q bit value
SENDDB - D bit value
SENDGFI - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)

S:DATAP (--)  
Transmits a data packet on the selected channel using values from the following variables:
LCN - Logical group and channel value
SENDQB - Q bit value
SENDDB - D bit value
SENDGFI - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
NVS - P(S) value
NVR - P(R) value
SENDM - More bit value
DATA-BUF-ID - Indication of which data buffer to use in user data field
  10 = buffer created with SENDD command (maximum length is 80 characters)
  11 = echo of received data packets
  Any other value - DATA-BUFFER which contains the default data field or data field created with DEFINE_DATA command (maximum 4100 characters)
LCNSIZE - Length of the data field to send when DATA-BUFFER is used
MAX_LENGTH - Maximum frame length

S:RRP (--)  
Transmits an RR packet on the selected channel using values from the following variables:
LCN - Logical group and channel value
SENDQB - Q bit value
SENDDB - D bit value
SENDGFI - GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
NVR - P(R) value
S:RNRP (--)  
Transmits an RNR packet on the selected channel using values from the following variables:
- LCN: Logical group and channel value
- SENDQB: Q bit value
- SENDDB: D bit value
- SENDGFI: GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
- NVR: P(R) value

S:REJP (--)  
Transmits an REJ packet on the selected channel using values from the following variables:
- LCN: Logical group and channel value
- SENDQB: Q bit value
- SENDDB: D bit value
- SENDGFI: GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
- NVR: P(R) value

S:DIAG (--)  
Transmits a diagnostic packet on the selected channel using values from the following variables:
- LCN: Logical group and channel value
- SENDQB: Q bit value
- SENDDB: D bit value
- SENDGFI: GFI modulo value (0x10 for modulo 8, and 0x20 for modulo 128)
- DIAG: Diagnostic value
- DIAG-EXP: Diagnostic explanation field. The first byte is the length of the field followed by up to three bytes for the explanation field to be sent.
IDACOM has developed a comprehensive set of tools for the development of test scripts. These test scripts, written using the ITL language, control the operation of the ISDN Monitor/Emulation program.

For a complete explanation of the test manager and tools available, see the Programmer's Reference Manual.

This section reviews basic ITL components plus describes the protocol event and action commands specific to ISDN.

### 20.1 ITL Constructs

Following is a brief description of test manager constructs. For more details and examples, refer to the Programmer's Reference Manual.

**TCLR (--)**

Initializes the test manager. Any existing test suites already in memory are cleared. The current state is set to 0. All test scenarios should start with the TCLR command.

**STATE_INIT{ }STATE_INIT (number --)**

Brackets the execution sequence performed prior to entering a state. The initialization logic for a state is executed independently of how it was called.

This initialization procedure can be used for any state but is not compulsory. STATE_INIT{ must be preceded by the number of the state being initialized, eg. 0 STATE_INIT{.

The STATE_INIT{ }STATE_INIT clause is executed only once each time the state is entered from another state.

**STATE{ }STATE (number --)**

Brackets a state definition. STATE{ must be preceded by the number of the state. Valid values are 0 through 255. State 0 must be defined within an ITL program. If not, the test manager will not run the script. If multiple states are defined with the same number in the test script, the test manager uses the latest definition.

**ACTION{ }ACTION (flag --)**

Brackets the set of tasks, decisions, and outputs which execute once the expected event is received by the test manager. There must be at least one action defined for each expected event. The action is executed when the flag is true (non-zero).
NEW_STATE ( n -- )
Executes the initialization logic of the specified state (providing \texttt{STAT\_INIT{ } }) \texttt{STAT\_INIT} is defined) and establishes the state to be executed for the next event. Any remaining action code for the current state is then executed. It must be preceded with a valid state number and be inside the \texttt{ACTION{ } } \texttt{ACTION} brackets. This command is not mandatory if no state change is desired.

TM\_STOP ( -- )
Stops the execution of the test script. The test suite remains in memory and can be re-executed until another test script is loaded.

SEQ{ }SEQ ( number -- )
Brackets a definition of tasks and outputs which execute as part of the state machine action. SEQ{ } expects a single integer which is the sequence number. Up to 256 sequences are supported. Valid values are 0 through 255. The SEQ{ }SEQ partners are extremely useful when more than one action sequence calls the same tasks and outputs. The SEQ{ }SEQ definition is defined outside the \texttt{ACTION{ } } \texttt{ACTION} definition and then called by the \texttt{RUN\_SEQ} command.

This is an alternate mechanism to generate colon definitions. This mechanism causes the equivalent of a colon definition (now accessed via a numeric identifier) to be compiled into the test script dictionary rather than the user dictionary. Refer to the Programmer’s Reference Manual.

RUN\_SEQ ( number -- )
Executes a specified set of tasks defined in a SEQ{ }SEQ definition. It is called inside an \texttt{ACTION{ } } \texttt{ACTION} definition and must be preceded with a defined sequence number.

LOAD\_RETURN\_STATE ( number -- )
Permits the test script writer to program the equivalent of subroutine calls (used with \texttt{RETURN\_STATE}). LOAD\_RETURN\_STATE sets the state to which control is to be returned. LOAD\_RETURN\_STATE must be within the action field; nesting is not permitted.

RETURN\_STATE ( -- )
Returns control to the state specified by LOAD\_RETURN\_STATE from a state subroutine call.

NEW\_TM ( filename -- )
Loads and compiles the specified file and then starts the test manager at state 0. It can be included as part of the action field to load and execute another scenario.

TM\_INIT ( -- )
This DOER word is executed when a test script starts execution. It can be used to initialize variables for the script when using state 0 would not be appropriate (such as when using the Layer 3 Connection Multiplexor).

Example:
\begin{verbatim}
MAKE TM\_INIT
    #BASIC\_SET\_INTERFACE
    CLEAR\_ALL\_CNS
;
\end{verbatim}
20.2 Control Commands

STATE_ON ( -- )
Displays trace statements for every protocol or test manager state change. This is useful for debugging test scripts.

STATE_OFF ( -- )
Trace statements for protocol or test manager state changes are not displayed.

?BRI ( -- flag)
Returns true if the test script is running on a BRA application. This command is usually used with the compiler control commands (see the Programmer’s Reference Manual).

Example:

```
#IF ?BRI #IS_TRUE
   #BASIC SET_INTERFACE
#ELSE
   #PRIMARY SET_INTERFACE
#ENDIF
```

?PRI ( -- flag)
Returns true if the test script is running on a PRA application.

Example:

```
I#CHANNEL_ID ELEMENT>
   OCTET_3 INCLUDED

#IF ?PRI #IS_TRUE
   OCTET_3.2 INCLUDED
   OCTET_3.3 INCLUDED
   #NUMBER *COD ->CID_NUMBER/MAP !
   0 *COD ->CID_NUMBER !
#ENDIF
<ELEMENT
```

20.3 Event Recognition

During test script execution, any event received by the test manager is evaluated to determine if it matches the event-specifier of the first action within that state. If the evaluation does not return a true value, the following action clauses are evaluated in a sequential manner. Once an event evaluates true, the subsequent action clauses in that particular state are not examined.
Layer 1

?L1_EVENT (event ID -- flag)
Returns true if the specified layer 1 event has occurred. The layer 1 event identifiers are:

<table>
<thead>
<tr>
<th>Basic Rate Access</th>
<th>Primary Rate Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>R#ACTIVATE</td>
<td>R#NORMAL</td>
</tr>
<tr>
<td>R#DEACTIVATE</td>
<td>R#OOFALM</td>
</tr>
<tr>
<td>R#RI2ERR</td>
<td>R#REDALM</td>
</tr>
<tr>
<td>R#LSTFRM</td>
<td>R#LSTSIGL</td>
</tr>
<tr>
<td>R#RECOVERY</td>
<td>R#YELALM</td>
</tr>
<tr>
<td>R#UNDEFINED</td>
<td>R#LSTPLOCK</td>
</tr>
</tbody>
</table>

Example:
R#ACTIVATE ?L1_EVENT
ACTION[
  .
  code to be executed when S bus is activated
  .
]ACTION

Layer 2

?RX_FRAME (frame ID -- flag)
Returns true if the specified frame has been received. The frame identifiers are:

<table>
<thead>
<tr>
<th>R#I</th>
<th>R#RR</th>
<th>R#RNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>R#REJ</td>
<td>R#SABME</td>
<td>R#SABM</td>
</tr>
<tr>
<td>R#DISC</td>
<td>R#UA</td>
<td>R#DM</td>
</tr>
<tr>
<td>R#FRMR</td>
<td>R#UI</td>
<td>R#INVFRM</td>
</tr>
<tr>
<td>R#INVCRC</td>
<td>R#SHORTFRM</td>
<td>R#XID</td>
</tr>
</tbody>
</table>

Example:
R#SABME ?RX_FRAME
ACTION[
  .
  code to be executed when a SABME is received
  .
]ACTION

?COMMAND ( -- flag)
Returns true if a command frame is received.

?RESPONSE ( -- flag)
Returns true if a response frame is received.
?RX_MATCH (string -- flag)
Returns true if the user-defined string is found in the information field of the received information frame.

This is an anchored match, i.e. a byte-for-byte match starting at the first byte of the information field of the received information frame.

⚠️ NOTE
To accommodate "don't care" character positions, the question mark character for ASCII or hex 3F character can be used.

The following examples assume the received information field is 'idacom electronics'.

Example 1:
Match a string containing the first four characters in the field; a true flag is returned.
"idac" ?RX_MATCH

Example 2:
Match a string containing the last characters in the field; a false flag is returned.
"electronics" ?RX_MATCH

?RX_SEARCH (string -- flag)
Returns true if the user-defined string is found in the information field of the received information frame.

This is an unanchored match, i.e. searches for an exact match anywhere in the information field of the received information frame, regardless of position.

The following examples assume the received information field is 'idacom electronics'.

Example 1:
Match a string containing the first four characters in the field; a true flag is returned.
"idac" ?RX_SEARCH

Example 2:
Match a string containing the last characters in the field; a true flag is returned.
"electronics" ?RX_SEARCH

?ABORT (-- flag)
Returns true if an abort frame is received.

?CRC_ERROR (-- flag)
Returns true if a frame with a CRC error is received.
?L2_SERVICE (service primitive -- flag)
Returns true if the specified service primitive has been received. The layer 2 services primitives are:

- DL-EST#IND   data link established
- DL-EST#CONF  data link establish confirmation
- DL-REL#IND   data link release
- DL-REL#CONF  data link release confirmation
- DL-DATA#IND  data link sequenced data received
- DL-DATA#CONF data link confirm data transfer
- DL-UDATA#IND data link non-sequenced data received

Example:
DL-EST#IND   ?L2_SERVICE
ACTION[
   action to be performed if an establish indication has been received
]ACTION

?RX_DATA (-- flag)
Returns true if a layer 2 information or unnumbered information frame has been received with a SAPI = 0.

?FROM_NT (-- flag)
Returns true if a layer 2 frame was received from the network.

Layer 3

?L3_MSG (message identifier -- flag)
Returns true if the specified message has been received. Valid message identifiers are listed in each message set manual.

Example:
M#SETUP   ?L3_MSG
ACTION[
   code to be executed if a setup message is received
]ACTION

?L3_MATCH (string -- flag)
Returns true if the specified string is found in the target string of a layer 3 message.

This is an anchored match, i.e. a byte-for-byte match starting at the first byte of the received message.
?L3_SEARCH ( string -- flag )
Returns true if the specified string is found in the target string of a layer 3 message.

This is an unanchored match, i.e. searches for an exact match anywhere in the received frame, regardless of position.

---

**X.25 Layer 3**

If the received frame is an information frame, the packet type is stored in the PACKET-TYPE variable. The identifiers used to match received packets are listed in Table 20–1.

<table>
<thead>
<tr>
<th>Packet Identifiers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*DATAP</td>
<td>Data packet</td>
</tr>
<tr>
<td>R*RRP</td>
<td>Receive ready packet</td>
</tr>
<tr>
<td>R*RNRP</td>
<td>Receive not ready packet</td>
</tr>
<tr>
<td>R*REJP</td>
<td>Reject packet</td>
</tr>
<tr>
<td>R*CALLREQ</td>
<td>Call request or incoming call packet</td>
</tr>
<tr>
<td>R*CALLCON</td>
<td>Call connect or call accept packet</td>
</tr>
<tr>
<td>R*CLEARREQ</td>
<td>Clear request or clear indication packet</td>
</tr>
<tr>
<td>R*CLEARCONF</td>
<td>Clear confirm packet</td>
</tr>
<tr>
<td>R*INTREQ</td>
<td>Interrupt packet</td>
</tr>
<tr>
<td>R*INTCONF</td>
<td>Interrupt confirm packet</td>
</tr>
<tr>
<td>R*RESETREQ</td>
<td>Reset request or reset indication packet</td>
</tr>
<tr>
<td>R*RESETCONF</td>
<td>Reset confirm packet</td>
</tr>
<tr>
<td>R*RESTARTREQ</td>
<td>Restart request or restart indication packet</td>
</tr>
<tr>
<td>R*RESTARTCONF</td>
<td>Restart confirm packet</td>
</tr>
<tr>
<td>R*DIAGNOSTIC</td>
<td>Diagnostic packet</td>
</tr>
<tr>
<td>R*REGISTREQ</td>
<td>Registration request packet</td>
</tr>
<tr>
<td>R*REGISTCONF</td>
<td>Registration confirm packet</td>
</tr>
<tr>
<td>R*INVPKT</td>
<td>Invalid packet</td>
</tr>
</tbody>
</table>

**Table 20–1 Packet Identifiers**

⚠️ **NOTE**

*If the received frame does not contain a packet, the PACKET-TYPE variable contains 0.*

For a more complete list of decoded communication variables, refer to Section 19.2.
?RX (packet id #1...packet id#n
-- flag)
Returns true if one of the specified packets is received. See Table 20-1 for valid packet identifiers.

\begin{itemize}
\item \textbf{NOTE} \hspace{1cm}
\begin{itemize}
\item In the emulation, layer 3 state machines are executed automatically.
\end{itemize}
\end{itemize}

Example:
Look for reception of invalid frames or packets using the monitor. The user receives an audible alarm and a notice.

\begin{verbatim}
3 STATE{
  R*INVFRM R*INVPKT 2 ?RX
  ACTION{
    BEEP
    "Invalid frame or packet received." W.NOTICE
  }ACTION
}STATE
\end{verbatim}

Example:
Look for the reception of a call request packet using the emulation. The automatic emulation provides the response and the test manager proceeds to state 6.

\begin{verbatim}
5 STATE{
  R*CALLREQ 1 ?RX
  ACTION{
    6 NEW_STATE (Layer 3 state machine provides response)
  }ACTION
}STATE
\end{verbatim}

?RX_PACKET (packet id#1...packet id#n
-- flag)
Returns true if one of the specified packets is received. This command provides the same operation as the ?RX command. See ?RX for examples.

\begin{itemize}
\item \textbf{NOTE} \hspace{1cm}
\begin{itemize}
\item The operation of this command is different than in the X.25 Emulation program. For ISDN LAPD, the layer 3 emulation state machine is executed for packet events. For X.25 LAPB, the layer 3 emulation state machine is not executed for packet events. When operation of the layer 3 emulation state machine is to be blocked, it can be turned off with the L3_OFF command.
\end{itemize}
\end{itemize}
?RX_X25_DATA ( string -- flag )
Returns true if a user-defined character string is found in the data field of received packets.

This is an anchored match, i.e. a byte-for-byte match starting at the first byte of the data field of a received data packet.

\[ \text{NOTE} \]
\text{The } ?RX_X25\text{ DATA provides the same function as } ?RX\text{ DATA does in the X.25 LAPB Emulation.} \]

\[ \text{NOTE} \]
\text{To accommodate "don't care" character positions, the question mark character for ASCII or hex 3F character can be used. The specified string is limited to 80 characters. The received data field can be longer than the specified string.}

Example:
Search for the string 'IDACOM' starting at the second byte of a data packet. The first byte is a "don't care" and matches on any value.

```
?IDACOM ?RX_X25\\_DATA
```
or
```
X"3F49\text{A}4\text{1}4\text{3}4F4D" ?RX_X25\text{DATA}
```

\[ \text{WINDOW?} ( -- flag ) \]
Calculates the window for the selected logical channel. Returns 0 if the window is closed and 1 if the window is open.

Example:
Send a data packet while the window is open.

```
BEGIN
  WINDOW?
  ( Check LCN Data Window )
  WHILE DATA
  ( While window is open )
  REPEAT
  ( Repeat until window is closed )
```

IDACOM

ISDN Programmer's Manual
Timeout Detection

There are 128 user programmable timers available. Timers 101 through 128 can be used in the test manager. Timer 4 is the test manager wakeup timer. The following timers are used within the application and should not be started or stopped in a test script.

**TIMEOUT (flag)**

Returns true if any timer has expired.

Example:
In State 8, look for the expiration of any timer. The action is to display a trace statement.

```
8 STATE{
    TIMEOUT       ( Check for timeout of any timer )
    ACTION{
        T. "A Timer has expired." TCR
    }ACTION
}STATE

?TIMER (timer # flag)

Returns true if the specified timer has expired. Valid input parameters are timers 101 through 128.

Example:
In State 8, look for the expiration of timer 101. The action is to display a trace statement.

```
8 STATE{
    101 ?TIMER       ( Check for timeout of timer 101 )
    ACTION{
        T. "Timer 101 has expired." TCR
    }ACTION
}STATE
```
?WAKEUP ( -- flag )
Returns true if the wakeup timer has expired. The wakeup timer can be used to initiate action sequences immediately upon the test manager starting. Timer 4 is started for 100 milliseconds when the test manager is started after a WAKEUP_ON command has been issued. The default is WAKEUP_OFF.

Example:
In State 0, look for the expiration of the wakeup timer. The action is to prompt the user to press a function key, and then the test manager goes to State 1.

```plaintext
0 STATE{
  ?WAKEUP          ( Check for timeout of wakeup timer )
  ACTION[          
    T." To start the test, press U1." TCR
    1 NEW_STATE
  ] ACTION
} STATE
```

TIMER–NUMBER ( -- address )
Contains the number of the expired timer. Valid values are 1 through 128.

?CN TIMER ( timer identifier -- flag )
Returns true when the layer 3 connection simulation timer has expired. Refer to the 'Connection Timer Management' section on page 18–8 for more information on connection timers.

LCNTIMER ( -- address )
Contains the timer value for the selected LCN.

Example:
Detect a timeout on channel 50.

```plaintext
TIMEOUT
TIMER–NUMBER @ CH50 LCNTIMER @ =
AND
ACTION[
  ...]
] ACTION
```
or

```plaintext
CH50 LCNTIMER @ ?TIMER
ACTION[
  ...]
] ACTION
```

NOTE
The two previous examples provide identical actions. The ?TIMER example is a more efficient method of coding.
Function Key Detection

Refer to the Programmer's Reference Manual.

Interprocessor Mail Events

Refer to the Programmer's Reference Manual.

Wildcard Events

OTHER_EVENT ( -- true flag )

Returns true so that an ACTION{}ACTION statement can check for events that have not been explicitly programmed into an event field.

The actual event is held in the EVENT-TYPE variable. Event identifiers are:

- LAYER1# L.1 event in L1-ID*
- FRAME# L.2 frame in FRAME-ID
- TIME-OUT# timer number in TIMER-NUMBER
- FUNCTION-KEY# function key in KEY-NUMBER
- COMMAND_IND mail has been received

\[\text{NOTE}\]

OTHER_EVENT should be the last item in a list of actions in a state definition.

Example:
OTHER_EVENT
ACTION{

    default actions for this state

}

}ACTION
20.4 ISDN Actions

All of the general actions explained in the Programmer's Reference Manual are supported in the ISDN Monitor and Emulation.

Layer 1 Actions

The following emulation commands turn control leads on and off (WAN only).

<table>
<thead>
<tr>
<th>V.28/RS–232C Interface</th>
<th>OFF to ON</th>
<th>ON to OFF</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS_ON</td>
<td>RTS_OFF</td>
<td></td>
<td>Request to send</td>
</tr>
<tr>
<td>CTS_ON</td>
<td>CTS_OFF</td>
<td></td>
<td>Clear to send</td>
</tr>
<tr>
<td>DSR_ON</td>
<td>DSR_OFF</td>
<td></td>
<td>Data set ready</td>
</tr>
<tr>
<td>CD_ON</td>
<td>CD_OFF</td>
<td></td>
<td>Carrier detect</td>
</tr>
<tr>
<td>DTR_ON</td>
<td>DTR_OFF</td>
<td></td>
<td>Data terminal ready</td>
</tr>
<tr>
<td>SQ_ON</td>
<td>SQ_OFF</td>
<td></td>
<td>Signal quality</td>
</tr>
<tr>
<td>RI_ON</td>
<td>RI_OFF</td>
<td></td>
<td>Ring indicate</td>
</tr>
<tr>
<td>DRS_ON</td>
<td>DRS_OFF</td>
<td></td>
<td>Data signal rate select</td>
</tr>
</tbody>
</table>

Table 20–2 V.28/RS–232C Interface Lead Transitions

<table>
<thead>
<tr>
<th>V.35 Interface</th>
<th>OFF to ON</th>
<th>ON to OFF</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS_ON</td>
<td>RTS_OFF</td>
<td></td>
<td>Request to send</td>
</tr>
<tr>
<td>CTS_ON</td>
<td>CTS_OFF</td>
<td></td>
<td>Clear to send</td>
</tr>
<tr>
<td>DSR_ON</td>
<td>DSR_OFF</td>
<td></td>
<td>Data set ready</td>
</tr>
<tr>
<td>CD_ON</td>
<td>CD_OFF</td>
<td></td>
<td>Carrier detect</td>
</tr>
<tr>
<td>DTR_ON</td>
<td>DTR_OFF</td>
<td></td>
<td>Data terminal ready</td>
</tr>
<tr>
<td>RI_ON</td>
<td>RI_OFF</td>
<td></td>
<td>Ring indicate</td>
</tr>
</tbody>
</table>

Table 20–3 V.35 Interface Lead Transitions
Lapse Timers

There are 256 lapse timers, numbered from 1 to 256, that can be started and read at any time.

START_LAPSE_TIMER (timer number --)
Starts the specified timer.

MINUTES_ELAPSED (timer number -- minutes)
Returns the number of minutes that have elapsed since the timer had started.

SECONDS_ELAPSED (timer number -- seconds)
Returns the number of seconds that have elapsed since the timer had started.

MILLISECONDS_ELAPSED (timer number -- milliseconds)
Returns the number of milliseconds that have elapsed since the timer had started.

Example:
1 START_LAPSE_TIMER
1 MINUTES_ELAPSED
1 SECONDS_ELAPSED

(Starts lapse timer 1)
(Returns # of minutes since timer was started)
(Returns # of seconds since timer was started)

### Table 20-4 V.36/RS-449 Interface Lead Transitions

<table>
<thead>
<tr>
<th>OFF to ON</th>
<th>ON to OFF</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS_ON</td>
<td>RTS_OFF</td>
<td>Request to send</td>
</tr>
<tr>
<td>CTS_ON</td>
<td>CTS_OFF</td>
<td>Clear to send</td>
</tr>
<tr>
<td>DSR_ON</td>
<td>DSR_OFF</td>
<td>Data set ready</td>
</tr>
<tr>
<td>DTR_ON</td>
<td>DTR_OFF</td>
<td>Data terminal ready</td>
</tr>
<tr>
<td>RI_ON</td>
<td>RI_OFF</td>
<td>Calling indicator</td>
</tr>
<tr>
<td>DRS_ON</td>
<td>DRS_OFF</td>
<td>Data signal rate select</td>
</tr>
<tr>
<td>CD_ON</td>
<td>CD_OFF</td>
<td>Data channel received line signal</td>
</tr>
</tbody>
</table>

V.36/RS-449 Interface

There are 256 lapse timers, numbered from 1 to 256, that can be started and read at any time.
Interval Timers

There are 28 interval timers, numbered from 101 to 128, that are available to the user.

**START_TIMER** (timer/interval -- )

Starts the timer for the specified interval.

The timeout of the timer can be determined through the use of the ?TIMER command.

Example:

```
0 STATE_INIT{
  101 20 START_TIMER
}STATE_INIT
0 STATE{
  101 ?TIMER
  ACTION{
    .
    action to be performed if timer 101 expires
    .
  }ACTION
}STATE
```

**STOP_TIMER** (timer -- )

Stops the specified timer.

Using Buffers

IDACOM's test manager has 256 buffers available for creating customized frames. These buffers are numbered from 0 through 255 and can be created any size desired. However, the emulation can limit the number of bytes that can be transmitted.

A buffer consists of four bytes with values of 0, two bytes containing the length of the text, and the remaining bytes consisting of user-defined text.

```
0 0 0 0 Length 0 Text Bytes
```

```
Bytes 1 to 4  Byte 5 & 6
```

**Figure 20-1 Buffer Structure**

**NOTE**

All buffers are cleared when the TCLR command is issued. TCLR is usually the first command compiled when loading a test script.
There are three methods of moving text into a buffer.

Methods 1 and 2 automatically allocate memory for the specified text. Method 3 requires the user to allocate memory before moving text into the buffer. Use the TCLR command to clear all buffers.

Method 1

STRING->BUFFER ( string\buffer number -- )
Loads a quoted string into the specified buffer. The length is limited to 80 bytes if typing directly on the keyboard and 255 bytes if used within a test script. Either an ASCII or hex string can be specified. Valid buffer numbers are 0 through 255.

Example:
" IDACOM" 1 STRING->BUFFER ( ASCII text moved to Buffer #1 )
X" 0100100100434445" 2 STRING->BUFFER ( Hex string of 8 bytes moved to Buffer #2 )

Method 2

FILE->BUFFER ( filename\buffer number -- )
Transfers a text file into the specified buffer (for text greater than 80 bytes). The file is created using the Edit function available on the Home processor. At this time, only ASCII text can be created. The last character to be transferred should be followed immediately by a CTRL ‘p’ character in the file. This special character is displayed as a pilcrow (¶) character. The file is transferred into the buffer until the ASCII control ‘p’ character is found or until the end of the file.

Example:
Create a file with the name CUSTOM.F and transfer to Buffer #3.
" CUSTOM.F" 3 FILE->BUFFER

Method 3

The following commands should not be used with FILE->BUFFER or STRING->BUFFER.

ALLOC_BUFFER ( size \ buffer number -- flag )
Allocates memory for the specified buffer. ALLOC_BUFFER returns 0 if an error occurred, or 1 if correct.

⚠️ NOTE
ALLOC_BUFFER should not be used repetitively with the same buffer number in the same test script.

FILL_BUFFER ( data address \ size \ buffer number -- )
Moves data, of a specified size, into a buffer. Previous contents are overwritten.

APPEND_TO_BUFFER ( data address \ size \ buffer number -- )
Appends data, of a specified size, into a buffer.
CLEAR_BUFFER ( buffer number -- )
Stores a size of 0 in the buffer. CLEAR_BUFFER has no effect on the allocated memory defined with ALLOT_BUFFER.

Example:
0 VARIABLE tempstring 6 ALLOT
  " A TEST " tempstring $!
16 3 ALLOT_BUFFER
IF
tempstring 4+ 5 3 FILL_BUFFER
  " FAIL" COUNT 3 APPEND_TO_BUFFER
ENDIF

BUFFER ( buffer number -- address | 0 )
Returns the address of the first byte of the specified buffer. The buffer must have been previously created by FILE->BUFFER, STRING->BUFFER, or ALLOT_BUFFER. A '0' is returned when the buffer is not created or an invalid buffer number is specified. Valid buffer numbers are 0 through 255.

20.5 X.25 PLP Actions

Packets can be transmitted utilizing any of the commands described in Section 19.3.

NOTE
The test manager can be run with the layer 2 and layer 3 state machines running automatically or turned off.

Before sending a buffer, the text must first be stored in the buffer using the STRING->BUFFER or FILE->BUFFER commands. Once the text is in place, the buffer can be transmitted repetitively.

The actual size of the frame or packet sent is defined by the frame and packet size set on the Emulation Configuration 2 Menu or by the value stored in the N201-TX* variable and the =SIZE command.

BUFFER_SENDP ( buffer number -- )
Transmits the specified buffer as the packet layer carried by an information frame. Valid buffer numbers are 0 through 255.

Example:
Create the text in the buffer and then transmit the buffer.
X" 10010043445" 3 STRING->BUFFER
3 BUFFER_SENDP
  ( Create text )
  ( Send packet )
BUFFER_SEND( buffer number -- )
Transmits the specified buffer as the data field of a data packet. Valid buffer numbers are 0 through 255.

Example:
Create the text in the buffer and then transmit the buffer.
" IDACOM" 4 STRING->BUFFER ( Create text )
4 BUFFER_SEND ( Send data packet )

Example:
The text to be included is longer than 80 characters and is in a file named CUSTOM.F.
" CUSTOM.F" 5 FILE->BUFFER ( Put text in buffer )
5 BUFFER_SEND ( Send Data packet )

20.6 Layer 3 Connection Multiplexer

A connection multiplexer is available to support simulation of a number of simultaneous connections. The connection multiplexer maintains a separate test manager state for each connection.

When an event occurs, the connection for which it is destined is determined, on the basis of the call reference value of the received layer 3 message. This connection is made active, and the test manager state for this connection is loaded and becomes the current state for the test manager. When the test manager has finished executing, the test manager state is saved for that connection and is used the next time an event occurs for that connection. The test script should make use of the connection management commands which access or set data relevant to the active connection.

When an event occurs that is not associated with any connection, such as receiving a message with a new call reference value, the test manager is started in state 0. Thus, state 0 is used to initialize new connections.

When the layer 3 multiplexer is turned off, the test manager executes immediately in response to an event.

TM_MUX_ON ( -- )
Turns on the layer 3 connection multiplexer.

TM_MUX_OFF ( -- )
Turns off the layer 3 connection multiplexer.

=CN_TM_STATE ( test manager state -- )
Sets the test manager state for the currently selected connection.

⚠️ WARNING
Use this command with caution so as not to corrupt the multiplexing behavior of the simulation.

?CN_TM_STATE ( -- test manager state )
Returns the test manager state for the currently selected connection.
Use the following commands to determine the connection pertaining to a particular event, without using the layer 3 connection multiplexer. These commands also activate the indicated connection.

**TIMER_TO_CN ( -- CN | -1 )**
Returns the connection number corresponding to the last timer expiry, or -1 if the timer expiry does not correspond to any particular connection.

**DETERMINE_CN ( -- CN | -1 )**
Returns the number of the connection associated with the last layer 3 message decoded, or -1 if no connection is appropriate.

**CR_TO_CN ( CR value\CR flag\SAPI\CES -- CN | -1 )**
Determines the connection corresponding to the call reference value, flag, SAPI, and CES provided, or -1 if no connection is appropriate.
This section contains sample test scripts which are also supplied on disk, and can be loaded and run as described in the Programmer’s Reference Manual. These scripts illustrate ISDN functions using the following methods:

- Using hex strings to define transmit and expected receive values;
- Using message pools to define transmitted messages;
- Using the message builder to dynamically define transmitted messages;
- Defining new functions to use within the test script;
- Event recognition (both hex and special message/IE support functions).

Test scripts DSM_1 and DMS_2 are two versions of a single test script which illustrate the use of the Northern Telecom NT_S208-2 message set.

DMS_1 uses hex strings to send messages and search for IE’s. Therefore, it is not necessary to select the NT message set.

**NOTE**

Using hex strings to search for IE’s can be inaccurate and can result in an erroneous match.

DMS_2, however, uses the dynamic message builder to build/send messages, and standard commands to identify messages and search for IE’s. The NT_S208-2 message set must be selected. This approach is more modular, easier to understand, and less prone to error.

The following table summarizes some of the methods used in the test scripts contained in this section.

<table>
<thead>
<tr>
<th></th>
<th>DMS_1</th>
<th>DMS_2</th>
<th>AT&amp;T</th>
<th>1TR6</th>
<th>L3SIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex Strings</td>
<td>YES</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pool Messages</td>
<td></td>
<td></td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>Dynamic Msg Bldr</td>
<td>YES</td>
<td></td>
<td></td>
<td></td>
<td>YES</td>
</tr>
<tr>
<td>User Side</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Network Side</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Message Set</td>
<td>NA</td>
<td>NT_S208-2</td>
<td>ATT_5E6</td>
<td>1TR6_NSA</td>
<td>CCITT_1988</td>
</tr>
</tbody>
</table>
21.1 DMS_1.F

This test script emulates both sides of a telephone call setup and teardown utilizing stimulus signalling procedures. The emulation answers a call from one terminal and then places a call to the other terminal. Once the call has been answered, the B-Channels are connected in a cross-looped manner. The originating telephone is expected to use B1, while the answering telephone is assigned B2.

The test script uses default TEI values of 1 and 2 for the two telephones. These can be modified by the use by changing the AREI commands in the initialization code for state 0.

( File Title: DMS_1.F )
( Simulates a switch to let 2 stimulus mode terminals call each other on the same S-bus. )
( ----------------------------------------------------------- )

NTL_S1 ( Select NT stimulus message set )
( Define words only if they have not already been defined. )

#IFNOTDEF ORIG-CES

( These words are used to manage the two links assigned to each terminal. )
( When a call is originated, the current link is saved in the variable ORIG-CES. Thus, we can set the current link to the calling terminal with the CALLING_CES command and to the called terminal with the CALLED_CES command. Note that links 0 and 1 are used for the two terminals. )

0 VARIABLE ORIG-CES ( Link # of the call originator )

: CALLING_CES ( -- ) ( Set the link to the originating side )
   ORIG-CES @ SA
;

: CALLED_CES ( -- ) ( Set the link to the answering side )
   ORIG-CES @
   DOCASE
      CASE 0 [ 1 SA ]
      CASE 1 [ 0 SA ]
   ENDCASE
;
These words are used to collect digits from the Keypad IE. KP is a buffer area that may contain 13 digits, plus the string length. Digits are appended to the string with the +KP command. It is used by the ?KP command, which scans the received layer 3 message for the Keypad IE, retrieves the digit character and saves it, and returns a logical flag to indicate whether or not the digit was present.

0 VARIABLE KP 10 ALLOT ( buffer for 13 digits maximum )

: +KP ( char -- )
  KP C@ 1+
  DUP KP C!
  KP + C!

: ?KP ( -- flag )
  L3-POINTER @ L3-LENGTH @ X" 2C01" ?*SEARCH ( scan for Keypad IE )
  IF
    C@ +KP YES ( fetch character and save in buffer )
  ELSE
    DROP NO ( drop the address and indicate failure )
  ENDIF

These commands are defined simply for convenience.

: ?ON_HOOK ( -- flag ) ( true if On Hook received )
  M#INFO ?L3_MSG ( received INFO message )
  X" 380198" ?L3_SEARCH AND ( and Feature Activation 24 present? )

: ?CALL_REQ ( -- flag ) ( true if Call Request received )
  M#INFO ?L3_MSG ( received INFO message )
  X" 380181" ?L3_SEARCH AND ( and Feature Activation 1 present? )

: SEND_CONF ( -- ) ( confirmation reply to On Hook )
  X" 3A0101" ?L3_SEARCH ( B Chan Control IE to release B2? )
  IF
    ( reply with CAD=space, PI=disconnect, FI=[1,0], BCC=disc B2 )
    X" 08007B0C0280201E02839696390281003A0101" DL_DATA
  ELSE
    ( reply with CAD=space, PI=disconnect, FI=[1,0], BCC=disc B1 )
    X" 08007B0C0280201E02839696390281003A0100" DL_DATA
  ENDIF

#ENDIF
21-4
September 1990

TEST SCRIPTS

TCLR

( Initialize Test Manager )

0 STATE_INIT[
  0 SA 0 SAPI 1 ATEI
  1 SA 0 SAPI 2 ATEI
  PORT_BRID DRT_OFF DROUTING
  DROP
]STATE_INIT

0 STATE[
  ?ON_HOOK
  ACTION{
    SEND_CONF
    7 NEW_STATE
  }ACTION

  ?CALL_REQ
  ACTION{
    ( save origination Connection Endpoint Suffix )
    CES* @ ORIG-CES !

    ( reply with PI=dialing, SIG=dial tone, FI=[1,1], BCC=conn B1 )
    X'08007B1E02829534010096390281013A0102H' DL_DATA

    ( route the B1 channel to a dial tone )
    PORT_BRID DIAL TONE_TYPE
    PORT_BRID VOICE BCHAN1 BCHAN_SRC
    PORT_BRID DRT_SERIAL DROUTING DROP

    1 NEW_STATE
  }ACTION
]STATE

1 STATE_INIT[
  0 KP C!
]STATE_INIT

1 STATE[
  ?ON_HOOK
  ACTION{
    SEND_CONF
    7 NEW_STATE
  }ACTION

ISDN Programmer's Manual

IDACOM
M/INFO ?L3_MSG
ACTION{
  ?KP
  ( First keypad value collected? )
  IF
    ( send SIG=tones off )
    X" 08007B34013F" DL_DATA
    ( send PI=dialing, SIG=dial tone, FI=[1,1] )
    X" 08007B1E0282953401009639028101" DL_DATA
    ( turn off tones )
    PORT_BRID OFF TONE_TYPE
  2 NEW_STATE
  ENDF
}ACTION
}STATE

2 STATE{
  ?ON_HOOK
  ACTION{
    SEND_CONF
    7 NEW_STATE
  }ACTION

  M/INFO ?L3_MSG
  ACTION{
    ?KP
    ( Second keypad value collected? )
    IF
      ( send SIG=tones off )
      X" 08007B34013F" DL_DATA
      3 NEW_STATE
    ENDF
  }ACTION
}STATE

3 STATE{
  ?ON_HOOK
  ACTION{
    SEND_CONF
    7 NEW_STATE
  }ACTION
M#INFO ?L3_MSG
ACTION{
  ?KP
  IF
    KP C@ 7 =
    IF
      ( send PI=call proceeding )
      X" 08007B1E028292" DL_DATA
      ( change link to destination terminal )
      CALLED_CES
      ( send SIG=incoming call, OAD="4624545", FI=[1,3] )
      X" 08007B3401406C0880343632343534359639028103" DL_DATA
      ( simulate switching delay with 1s timeout )
      101 10 START_TIMER
    4 NEW_STATE
  ENDF
  ENDIF
}ACTION
}STATE

4 STATE{
  ?ON_HOOK
  ACTION{
    SEND_CONF
    7 NEW_STATE
  }ACTION

  101 ?TIMER
  ACTION{
    ( change link to origination terminal )
    CALLING_CES
    ( send PI=call alerting, SIG=audible ring )
    X" 08007B1E028293340101" DL_DATA
    ( change tone type to audible ring )
    PORT_BRID AUD_RING TONE_TYPE
    5 NEW_STATE
  }ACTION
}STATE
5 STATE{
  ?CALL_REQ
  ACTION{
    ( send CAD="4624545", PI=call connected, SIG=alerting off,
      FI=[1,1], BCC=connect B2 )
    X" 08007B0C08830343632343534351E02829434014F96390281013A0103" DL_DATA
    ( turn off tone )
    PORT_BRID OFF TONE_TYPE
    ( change link to origination terminal )
    CALLING_CES
    ( send CAD=last 4 digits of KP, SIG=tones off )
    X" 08007B0C058030303034013F" DIALOG
    DUP KP 4 + SWAP 7 + 4 CMOVE
    DL_DATA
    ( send PI=call connected )
    X" 08007B1E028294" DL_DATA
    ( cross-loop B channels to connect call )
    PORT_BRID BCHAN1 BCHAN2 BCHAN_SRC
    PORT_BRID BCHAN2 BCHAN1 BCHAN_SRC
  } ACTION
  ?ON_HOOK
  ACTION{
    SEND_CONF
    6 NEW_STATE
  } ACTION
} STATE

6 STATE{
  ?ON_HOOK
  ACTION{
    SEND_CONF
    7 NEW_STATE
  } ACTION
} STATE

7 STATE{
  ?ON_HOOK
  ACTION{
    SEND_CONF
    0 NEW_STATE
  } ACTION
} STATE
21.2 DMS_2

DMS_2a.F

```plaintext
DMS_2a.F

{ ---------------
( File Title: DMS_2a.F )
( )
( Simulates a switch to let 2 stimulus mode terminals call )
( each other on the same S-bus. )
( )
{ ---------------

TCLR
  " NT_S208-2" LOAD_MESSAGE_SET
  0 =CR_LENGTH

( Define words only if they have not already been defined. )
#IPNOTDEF ORIG-CES

( These words are used to manage the two links assigned to each terminal. )
( When a call is originated, the current link is saved in the variable )
( ORIG-CES. Thus, we can set the current link to the calling terminal with )
( the CALLING_CES command and to the called terminal with the CALLED_CES )
( command. Note that links 0 and 1 are used for the two terminals. )

0 VARIABLE ORIG-CES
  : CALLING_CES  ( -- )
     ORIG-CES @ SA
;

: CALLED_CES  ( -- )
     ORIG-CES @ DOCASE
     CASE 0 { 1 SA }
     CASE 1 { 0 SA }
     ENDCASE
;
```

ISDN Programmer's Manual
These words are used to collect digits from the Keypad IE. KP is a buffer area that may contain 13 digits, plus the string length. Digits are appended to the string with the +KP command. It is used by the ?KP command, which scans the received layer 3 message for the Keypad IE, retrieves the digit character and saves it, and returns a logical flag to indicate whether or not the digit was present.

```
0 VARIABLE KP 10 ALLOT
   ( buffer for 13 digits maximum )

:+KP ( char -- )
   KP C@ 1+  ( add 1 to string length )
   DUP KP C!  ( store new string length )
   KP + C!  ( store character at end of string )
;

:?KP ( -- flag )
   I#KEYPAD 1 ?L3_IE
   IF
      *DEC ->KNTS_KEYPAD DUP C@ 1 =
      IF
         1 + C@ +KP YES  ( fetch character and save in buffer )
      ELSE
         DROP NO
      THEN
   ELSE
      DROP NO
   ENDIF
   ( drop the address and indicate failure )
;

( These commands are defined simply for convenience. )

( number -- flag )
:-FEATURE
   I#NTL_FEAT_ACT 1 ?L3_IE
   IF
      I#NTL_FEAT_ACT OCTET_3 ?L3_OCTET
      IF
         *DEC ->FA_VALUE @ =
      ELSE
         DROP 0
      THEN
   ELSE
      DROP 0
   THEN
;
```
: ?ON_HOOK  ( -- flag )  ( true if On Hook received )
   M#INFO ?L3_MSG  ( received INFO message )
      IF
         24 ?FEATURE  ( and Feature Activation 24 present? )
      ELSE
         0
      THEN

: ?CALL_REQ  ( -- flag )  ( true if Call Request received )
   M#INFO ?L3_MSG  ( received INFO message )
      IF
         1 ?FEATURE  ( and Feature Activation 1 present? )
      ELSE
         0
      THEN

   I#B-CHAN_CONTR 1 ?L3_IE
      IF
         I#B-CHAN_CONTR OCTET_3 ?L3_OCTET
            IF
               *DEC \BCC_PARAM @ #1 =
            ELSE
               0
            THEN
         ELSE
            0
         THEN

: SEND_CONF  ( -- )  ( confirmation reply to On Hook )
   ?REL_B2
      IF
         ( reply with CAD=space, PI=disconnect, FI=[1,0], BCC=disc B2 )
         M#INFO MESSAGE>
           I#CONN_ADDR DUP ELEMENT>
              OCTET_3 INCLUDED
                 0  *COD ->CA_ADDR_TYPE !
                 0  *COD ->CA_NUMBER/ADDR !
              OCTET_4 INCLUDED
                 " " *COD ->CA_ADDR_DIGIT !STRING
           <ELEMENT
I#PROGRESS_IND DUP ELEMENT> 
OCTET_3 INCLUDED 
 #CCITT 
 #TRANSIT 
 OCTET_4 INCLUDED 
 #DISCONNECT 
 <ELEMENT

I#NTL_FEAT_IND DUP ELEMENT> 
OCTET_3 INCLUDED 
 1 
 OCTET_4 INCLUDED 
 0 
 <ELEMENT

I#B-CHAN_CONTR DUP ELEMENT> 
OCTET_3 INCLUDED 
 #1 
 *COD ->BCC_PARAM !
 <ELEMENT

<SEND

ELSE 
( reply with CAD=space, PI=disconnect, FI=[1,0], BCC=disc B1 )

M/INFO MESSAGE>

I#CONN_ADDR DUP ELEMENT> 
OCTET_3 INCLUDED 
 0 *COD ->CA_ADDR_TYPE ! 
 0 
 OCTET_4 INCLUDED 
 - - 
 *COD ->CA_ADDR_DIGIT !STRING
 <ELEMENT

I#PROGRESS_IND DUP ELEMENT> 
OCTET_3 INCLUDED 
 #CCITT 
 #TRANSIT 
 OCTET_4 INCLUDED 
 #DISCONNECT 
 <ELEMENT

I#NTL_FEAT_IND DUP ELEMENT> 
OCTET_3 INCLUDED 
 1 
 OCTET_4 INCLUDED 
 0 
 <ELEMENT

IDACOM ISDN Programmer's Manual
I\#B-CHAN_CONTR DUP ELEMENT>
   OCTET_3 INCLUDED
       #0
   <ELEMENT
       *COD ->BCC_PARAM !
   <ELEMENT

SEND
ENDIF
;
#ENDIF

" DMS_2b.F" EXECF
DMS_2b.F

( File Title:  DMS_2b.F
( Description:  Continuation of DMS2a.F test script
( -----------------------------------------------

0 STATE_INIT{
    0 SA 0 SAPI 1 ATEI  ( Link 0:  SAPI 0 and TEI 1 )
    1 SA 0 SAPI 2 ATEI  ( Link 1:  SAPI 0 and TEI 2 )
    PORT_BRID DRT_OFF DROUTING  ( Set routing of B Channels to "ignore" )
    DROP  ( Ignore flag from DROUTING )
}STATE_INIT

0 STATE{
    ?ON_HOOK
    ACTION{
        SEND_CONF
        7 NEW_STATE
    }ACTION

    ?CALL_REQ
    ACTION{
        ( save origination Connection Endpoint Suffix )
        CES* @ ORIG-CES !

        ( reply with PI=dialing, SIG=dial tone; FI=[1,1], BCC=conn B1 )
        M#INFO MESSAGE>

        I#PROGRESS_IND DUP ELEMENT>
        OCTET_3 INCLUDED
        #CCITT  *COD ->PI_CODING_STANDARD !
        #LOCAL_PUBLIC  *COD ->PI_LOCATION !
        OCTET_4 INCLUDED
        #DIALING  *COD ->PI_DESCRIPTION !
        <ELEMENT

        I#SIGNAL DUP ELEMENT>
        OCTET_3 INCLUDED
        #DIAL_ON  *COD ->SI_VALUE !
        <ELEMENT

        I#NTL_FEAT_IND DUP ELEMENT>
        OCTET_3 INCLUDED
        1  *COD ->FI_VALUE !
        OCTET_4 INCLUDED
        #IDLE  *COD ->FI_STATE_PARAM !
        <ELEMENT

IDACOM  ISDN Programmer's Manual
I#B-CHAN_CTRNL DUP ELEMENT>

OCTET_3 INCLUDED

#2

*COD -> BCC_PARAM !

<SEND

( route the B1 channel to a dial tone )

PORT_BRID DIAL TONE_TYPE
PORT_BRID VOICE BCHAN1 BCHAN_SRC
PORT_BRID DRT_SERIAL DROUTING DROP

1 NEW_STATE
}ACTION
}STATE

1 STATE_INIT{

0 KP C!
}STATE_INIT

1 STATE{

?ON_HOOK
ACTION{

SEND_CONF

7 NEW_STATE
}ACTION

M#INFO ?L3_MSG
ACTION{

?KP
IF

( send SIG=tones off )

M#INFO MESSAGE>

I#SIGNAL DUP ELEMENT>

OCTET_3 INCLUDED

#TONES_OFF

*COD -> SI_VALUE !

<SEND

( send PI=dialing, SIG=dial tone, FI=[1,1] )

M#INFO MESSAGE>

I#PROGRESS_IND DUP ELEMENT>

OCTET_3 INCLUDED

#CCITT

#LOCAL_PUBLIC

OCTET_4 INCLUDED

#DIALING

*COD -> PI_CODING_STANDARD !

*COD -> PI_LOCATION !

*COD -> PI_DESCRIPTION !

<ELEMENT

<SEND

( Initialize keypad string length to zero )

( First keypad value collected? )

( send SIG=tones off )

( send PI=dialing, SIG=dial tone, FI=[1,1] )
I#SIGNAL DUP ELEMENT>
OCTET_3 INCLUDED
#DIAL_ON
<SIGNAL
*COD ->SI_VALUE !
</ELEMENT

I#NTL_FEAT_IND DUP ELEMENT>
OCTET_3 INCLUDED
1
OCTET_4 INCLUDED
#IDLE
<SIGNAL
*COD ->FI_VALUE !
</ELEMENT

<SIGNAL
*COD ->FI_STATE_PARAM !
</ELEMENT

<SEND
( turn off tones )
PORT_BRID OFF TONE_TYPE
2 NEW_STATE
ENDIF
)ACTION
)STATE

2 STATE{
?ON_HOOK
ACTION{
SEND_CONF
7 NEW_STATE
)ACTION
M#INFO ?L3_MSG
ACTION{
?KP
IF
( send SIG=tones off )
M#INFO MESSAGE>

I#SIGNAL DUP ELEMENT>
OCTET_3 INCLUDED
#TONES_OFF
)SIGNAL
*COD ->SI_VALUE !
</ELEMENT

<SEND
3 NEW_STATE
ENDIF
)ACTION
)STATE
3 STATE{
  ?ON_HOOK
  ACTION{
    SEND_CONF
    7 NEW_STATE
  }ACTION

  M/INFO ?L3_MSG
  ACTION{
    ?KP
    IF
    KP C@ 7 =
    IF
      ( Additional keypad value collected? )
      M/INFO MESSAGE>

      I#PROGRESS_IND DUP ELEMENT>
      OCTET_3 INCLUDED
      #CCITT *COD ->PI_CODING_STANDARD !
      #LOCAL_PUBLIC *COD ->PI_LOCATION !
      OCTET_4 INCLUDED
      #CALL_PROCEEDING *COD ->PI_DESCRIPTION !
          <ELEMENT

          <SEND

          ( change link to destination terminal )
          CALLED_CES

          ( send SIG= incoming call, OAD="4624545", FI=[1,3] )
          M/INFO MESSAGE>

          I#SIGNAL DUP ELEMENT>
          OCTET_3 INCLUDED
          #ALERTING_ON_0 *COD ->SI_VALUE !
              <ELEMENT

          I#ORIG_ADDR DUP ELEMENT>
          OCTET_3 INCLUDED
          0 *COD ->OA_ADDR_TYPE !
          0 *COD ->OA_NUMBER/ADDR !
          OCTET_4 INCLUDED
          " 4624545" *COD ->OA_ADDR_DIGIT !STRING
              <ELEMENT

          I#NTL_FEAT_IND DUP ELEMENT>
          OCTET_3 INCLUDED
          1 *COD ->FI_VALUE !
          OCTET_4 INCLUDED
          #HELD *COD ->FI_STATE_PARAM !
              <ELEMENT
<SEND

( simulate switching delay with 1s timeout )
101 10 START_TIMER

4 NEW_STATE
ENDIF
ENDIF
}

}ACTION

}STATE

4 STATE{
  ?ON_HOOK
  ACTION{
    SEND_CONF
    7 NEW_STATE
  }
}

101 ?TIMER
ACTION{
  ( change link to origination terminal )
  CALLING_CES

  ( send PI=call alerting, SIG=audible ring )
  M#INFO MESSAGE>

    I#PROGRESS_IND DUP ELEMENT>
    OCTET_3 INCLUDED
      #CCITT
      #LOCAL_PUBLIC
    OCTET_4 INCLUDED
      #ALERTING_DEST
    <ELEMENT

    I#SIGNAL DUP ELEMENT>
    OCTET_3 INCLUDED
      #RING_BACK_ON
    <ELEMENT

  <SEND

  ( change tone type to audible ring )
  PORT_BRID AUD_RING TONE_TYPE

  5 NEW_STATE
  }
}

}ACTION

}STATE
5 STATE{
  ?CALL_REQ
  ACTION{
    ( send CAD=“4624545”, PI=call connected, SIG=alerting off,
      FI=[1,1], BCC=connect B2 )
    M#INFO MESSAGE>

    I#CONN_ADDR DUP ELEMENT>
    OCTET_3 INCLUDED
      0
      0
    OCTET_4 INCLUDED
      “4624545”

    I#PROGRESS_IND DUP ELEMENT>
    OCTET_3 INCLUDED
      #CCITT
      #LOCAL_PUBLIC
    OCTET_4 INCLUDED
      #DEST_CONNECTED

    I#SIGNAL DUP ELEMENT>
    OCTET_3 INCLUDED
      #ALERTING_OFF

    I#NTL_FEAT_IND DUP ELEMENT>
    OCTET_3 INCLUDED
      1
    OCTET_4 INCLUDED
      #IDLE

    I#B-CHAN_CONTR DUP ELEMENT>
    OCTET_3 INCLUDED
      3

    <SEND>
    ( turn off tone )
    PORT_BRID OFF TONE_TYPE

    ( change link to origination terminal )
    CALLING_CES

    ( send CAD=last 4 digits of KP, SIG=tones off )
    M#INFO MESSAGE>
I#CONN_ADDR DUP ELEMENT>
  OCTET_3 INCLUDED
  0               *COD ->CA_ADDR_TYPE !
  0               *COD ->CA_NUMBER/ADDR !
  OCTET_4 INCLUDED
  KP 4 + ( address of the last 4 digits )
  *COD ->CA_ADDR_DIGIT DUP 4 SWAP C! ( store the length )
  1 + 4 CMOVE ( copy the string )
</ELEMENT>

I#SIGNAL DUP ELEMENT>
  OCTET_3 INCLUDED
  #TONES_OFF            *COD ->SI_VALUE !
</ELEMENT>

<SEND
( send PI=call connected )
M#INFO MESSAGE>

I#PROGRESS_IND DUP ELEMENT>
  OCTET_3 INCLUDED
  #CCITT                *COD ->PI_CODING_STANDARD !
  #LOCAL_PUBLIC        *COD ->PI_LOCATION !
  OCTET_4 INCLUDED
  #DEST_CONNECTED      *COD ->PI_DESCRIPTION !
</ELEMENT>

<SEND
( cross-loop B channels to connect call )
PORT_BRID BCHAN1 BCHAN2 BCHAN_SRC
PORT_BRID BCHAN2 BCHAN1 BCHAN_SRC

  6 NEW_STATE
)ACTION

?ON_HOOK
ACTION{
  SEND_CONF
  6 NEW_STATE
}ACTION
)STATE

6 STATE{
  ?ON_HOOK
  ACTION{
    SEND_CONF
    7 NEW_STATE
  }ACTION
}STATE
7 STATE{
  ?ON_HOOK
  ACTION{
    SEND_CONF
    0 NEW_STATE
  }ACTION
}STATE
21.3 ATT_NET.F

The ATT_NET.F test script simulates a network for handling calls made to a AT&T 7506 telephone. This phone also has a data link available for SAPI 16 communications. The network manages both links, but only handles call setup procedures on the SAPI 0 signalling link. The network initializes the telephone by powering the S/T Bus. When a call is established from the user, the network supplies a dial tone and an audible ring when the four digits have been collected. The voice connector on the testers connector module can be used for a handset. This script makes use of the message pool method to send messages. A pool file, ATT.P, is automatically loaded.

```plaintext
0 STATE_INIT{
    NTWK_EMUL
    0 SA 0 SAPI 3 ATEI
    1 SA 0 SAPI
    1 RX-CALLREF !
    1 TX-CALLREF !
    PORT_BRID PS1_OFF SET_PS
    PORT_BRID VOICE BCHAN1 BCHAN_SRC
    PORT_BRID DRT_OFF DROUTING DROP
    CLEAR.Keys
    "PS_On" 1 LABEL_KEY
    "TestKeys" SET_CURR_TOPIC
}STATE_INIT

0 STATE{
    UF1 ?KEY
    ACTION{
        PORT_BRID PS1+ SET_PS
        1 NEW_STATE
    }ACTION
}STATE

1 STATE_INIT{
    PORT_BRID DRT_OFF DROUTING DROP
    "MakeCall" 1 LABEL_KEY
}STATE_INIT
```

IDACOM

ISDN Programmer's Manual
1 STATE{
  M#SETUP ?L3_MSG    ( incoming call )
  ACTION{
    1 CLEAR_KEY     ( clear "MakeCall" label )
    $MSG-CRVALUE @ RX-CALLREF !    ( use the same Call. Ref. value )
    DEST_SIDE      ( indicate Call. Ref. flag is #DEST )
    " SETUP_ACK" POOL ALTER_CR SEND_I    ( acknowledge incoming SETUP )
    PORT_BRID DIAL TONE_TYPE    ( set codec to dial tone )
    PORT_BRID DRT_SERIAL DROUTING DROP    ( connect B channel )
    2 NEW_STATE    ( wait for first keypad digit )
  }ACTION

  UFl ?KEY    ( user wishes to initiate call )
  ACTION{
    1 CLEAR_KEY     ( erase "MakeCall" label )
    ORIG_SIDE      ( indicate Call. Ref. flag is #ORIG )
    1 SA           ( ensure the signalling link is used )
    BROADCAST      ( UI to be sent as TEI 127 )
    " SETUP" POOL ALTER_CR SEND_UI    ( send SETUP )
    PORT_BRID OFF TONE_TYPE    ( turn tones off )
    6 NEW_STATE    ( wait for TE to connect )
  }ACTION
}

2 STATE{
  M#INFO ?L3_MSG    ( Keypad IE in INFO message received )
  I#KEYPAD 1 ?L3_IE AND
  ACTION{
    1 COUNTER1 !    ( first key received )
    " TONES_OFF" POOL ALTER_CR SEND_I    ( send Tones Off signal )
    PORT_BRID OFF TONE_TYPE    ( turn tones off )
    3 NEW_STATE    ( wait for next key )
  }ACTION
}

3 STATE{
  M#INFO ?L3_MSG    ( Keypad IE in INFO message received )
  I#KEYPAD 1 ?L3_IE AND
  ACTION{
    1 COUNTER1 +!    ( additional key received )
    COUNTER1 @ 4 =    ( 4 digits received? )
    IF
      " CALL_PROC" POOL ALTER_CR SEND_I    ( send CALL PROCeeding )
    101 10 START_TIMER    ( timer simulates switching delay )
    4 NEW_STATE    ( wait for timer to expire )
    ENDIF
  }ACTION
}

ISDN Programmer's Manual

IDACOM
4 STATE{
  101 ?TIMER ("switching delay" timer expired)
  ACTION{
    1 SA (ensure the signalling link is used)
    " ALERT" POOL ALTER_CR SEND_I (send ALERTing at Destination)
    101 30 START_TIMER (simulate delay for answer at far end)
    PORT_BRID AUD_RING TONE_TYPE (change tone to Audible Ring)
    5 NEW_STATE (wait for timer to expire)
  }ACTION
}STATE

5 STATE{
  101 ?TIMER ("answer delay" timer expired)
  ACTION{
    1 SA (ensure the signalling link is used)
    " CONN" POOL ALTER_CR SEND_I (send CONNECTed)
    PORT_BRID OFF TONE_TYPE (turn off tones)
    7 NEW_STATE (wait for disconnect)
  }ACTION
}STATE

6 STATE{
  M#CONN ?L3_MSG (Connect from TE)
  ACTION{
    PORT_BRID DRT_SERIAL DROUTING DROP (connect B channel)
    " CONN_ACK" POOL ALTER_CR SEND_I (send CONNECTION ACKnowledge)
    7 NEW_STATE (wait for disconnect)
  }ACTION
}STATE

7 STATE_INIT{
  " HangUp" 1 LABEL_KEY (label first key)
}STATE_INIT

7 STATE{
  M#DISC ?L3_MSG (TE goes On Hook and disconnects)
  ACTION{
    " REL" POOL ALTER_CR SEND_I (send RELEASE)
    1 NEW_STATE (wait for origination of next call)
  }ACTION

  UF1 ?KEY (user wishes to Disconnect call)
  ACTION{
    1 CLEAR_KEY (erase "HangUp" label)
    1 SA (ensure the signalling link is used)
    " DISC" POOL ALTER_CR SEND_I (send DISConnect)
    8 NEW_STATE (wait for TE to release)
  }ACTION
}STATE
8 STATE[
  M#REL ?L3_MSG
  ( Release from TE )
  ACTION[
    " REL_CoM" POOL ALTER_CR SEND_I
    ( send RELease COMplete )
    1 NEW_STATE
    ( wait for origination of next call )
  ] ACTION
] STATE

Message Pool Descriptions

There are ten pool entries in the ATT.P message pool file, as described in the following list. Each pool entry is shown in hexadecimal and complete formats.

1. SETUP_ACK

08 01 81 0D

18 01 89
34 01 00
96

PD = Q.931  CR = 0x01  Dest SETUP ACKnowledge  Var. = ATT_5E6

<table>
<thead>
<tr>
<th></th>
<th>INFORMATION ELEMENT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00011000</td>
<td>CHANNEL IDentification</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>00000001</td>
<td>IE length : 1 octets</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>01</td>
<td>Extension bit : not continued</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0-</td>
<td>Interface ident. : Interface implicitly identified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>---0-</td>
<td>Interface type : Basic Rate Interface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>---0-</td>
<td>Spare :</td>
<td></td>
</tr>
<tr>
<td></td>
<td>------1-</td>
<td>Preferred/Exclusive : exclusive; only this ch. acceptable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>------0--</td>
<td>D-channel indicator : The channel is not the D-channel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>------01</td>
<td>Info. chan. sel. : B1-channel</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>INFORMATION ELEMENT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00110100</td>
<td>SIGNAL</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>00000001</td>
<td>IE length : 1 octets</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>00000000</td>
<td>Signal value : Dial tone on</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1001-</td>
<td>INFORMATION ELEMENT : SHIFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0---</td>
<td>Shift type : locking</td>
<td></td>
</tr>
<tr>
<td></td>
<td>------110</td>
<td>New codeset ident. : local service network specific IEs</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>INFORMATION ELEMENT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00111100</td>
<td>DISPLAY FIELD</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>00000010</td>
<td>IE length : 4 octets</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0001-</td>
<td>Display Mode : Normal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0001</td>
<td>Submode : Direct</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>000---</td>
<td>Spare :</td>
<td></td>
</tr>
<tr>
<td></td>
<td>------0001</td>
<td>Display Field Type : Call Appearance ID</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>**********</td>
<td>Display information : [ 1]</td>
<td></td>
</tr>
</tbody>
</table>
2. TONES_OFF

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>08 01 01 7B</td>
<td></td>
<td>34 01 3F</td>
</tr>
</tbody>
</table>

PD = Q.931 CR = 0X01 Orig INFORMATION Var. = ATT_5E6
1 00110100 INFORMATION ELEMENT : SIGNAL
2 00000001 IE length : 1 octets
3 00111111 Signal value : Tones off

3. CALL_PROC

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>08 01 81 02</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PD = Q.931 CR = 0X01 Dest CALL PROCeeding Var. = ATT_5E6

4. ALERT

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>08 01 81 01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PD = Q.931 CR = 0X01 Dest ALERTing Var. = ATT_5E6

5. REL

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>08 01 81 4D</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PD = Q.931 CR = 0X01 Dest RELEASE Var. = ATT_5E6
1 00001000 INFORMATION ELEMENT : CAUSE
2 00000010 IE length : 2 octets
3 1------ Extension bit : not continued
   -00---- Coding standard : CCITT standardized in Q.931
   ----0---- Spare : 
   ----0010 Location : Public Network Serving Local User
4 1------ Extension bit : not continued
   -001---- Class : Normal event
   ----0000 Cause value : 16 Normal, Clearing
1 00110100 INFORMATION ELEMENT : SIGNAL
2 00000001 IE length : 1 octets
3 01001111 Signal value : Alerting off
1 1001---- INFORMATION ELEMENT : SHIFT
   -----0---- Shift type : locking
   -----110 New codeset ident. : local service network specific IEs
1 00100010 INFORMATION ELEMENT : SELECTed CALL APPEARance
2 00000001 IE length : 1 octets
3 00000001 Button Nr/Call App : 1
6. SETUP

08 01 01 05
04 03 80 90 A2
18 01 89
34 01 40
96
22 01 01
25 01 01

PD = Q.931 CR = 0X01 Orig SETUP Var. = ATT_5E6

1 0000100 INFORMATION ELEMENT : BEARER CAPability
2 0000011 IE length : 3 octets
3 1------ Extension bit : not continued
   -00----- Coding standard : CCITT standardized in Q.931
   ---0000 Info. trans. cap. : speech
4 1------ Extension bit : not continued
   -00----- Transfer mode : circuit mode
   ---1000 Info. transfer rate : 64 kbit/s
5 1------ Extension bit : not continued
   -01----- Layer ident. : User information layer 1 protocol
   ---0010 Layer 1 protocol : VALUE NOT SPECIFIED BY AT&T
1 00011000 INFORMATION ELEMENT : CHANNEL IDentification
2 00000001 IE length : 1 octets
3 1------ Extension bit : not continued
   -0----- Interface ident. : Interface implicitly identified
   --0----- Interface type : Basic Rate Interface
   ---0---- Spare : 
   -----1--- Preferred/Exclusive : exclusive; only this ch. acceptable
   -----0---- D-channel indicator : The channel is not the D-channel
   -------01 Info. chan. sel. : Bl-channel
1 00110100 INFORMATION ELEMENT : SIGNAL
2 00000001 IE length : 1 octets
3 01000000 Signal value : Normal alerting
1 1001---- INFORMATION ELEMENT : SHIFT
   ----0--- Shift type : locking
   ------110 New codeset ident. : local service network specific IEs
1 00100010 INFORMATION ELEMENT : SELECTed CALL APPEARance
2 00000001 IE length : 1 octets
3 00000001 Button Nr/Call App : 1
1 00100101 INFORMATION ELEMENT : DESTination CALL APPEARance
2 00000001 IE length : 1 octets
3 00000001 Dest. call app. : 1
7. CONN_ACK

08 01 5A 0F
18 01 89
34 01 4F

PD = Q.931   CR = 0x5A    Orig CONNect ACKnowl    Var. = ATT_5E6
1  00011000 INFORMATION ELEMENT : CHANNEL IDentification
2  00000001 IE length : 1 octets
3  1----- Extension bit : not continued
   -0------ Interface ident. : Interface implicitly identified
   --0----- Interface type : Basic Rate Interface
   ----0--- Spare :
   -----1--- Preferred/Exclusive : exclusive; only this ch. acceptable
   ------0-- D-channel indicator : The channel is not the D-channel
1  00110100 INFORMATION ELEMENT : SIGNAL
2  00000001 IE length : 1 octets
3  01001111 Signal value : Alerting off

8. DISC

08 01 5A 45
08 02 80 90

PD = Q.931   CR = 0x5A    Orig DISConnect    Var. = ATT_5E6
1  00001000 INFORMATION ELEMENT : CAUSE
2  00000010 IE length : 2 octets
3  1----- Extension bit : not continued
   -00----- Coding standard : CCITT standardized in Q.931
   ----0--- Spare :
   ------00 Location : User
4  1------ Extension bit : not continued
   -001---- Class : Normal event
   ------000 Cause value : 16 Normal, Clearing
9. REL_COM

08 01 5A 5A
08 02 82 90
96
22 01 00

PD = Q.931  CR = 0x5A  Orig  RELlease COMPLETE  Var. = ATT_5E6

1  00001000 INFORMATION ELEMENT : CAUSE
2  00000010 IE length : 2 octets
3  1-------- Extension bit : not continued
   -00------ Coding standard : CCITT standardized in Q.931
   ----0---- Spare :
   ----0010 Location : Public Network Serving Local User
4  1-------- Extension bit : not continued
   -001---- Class : Normal event
   ----0000 Cause value : 16 Normal, Clearing
1  1001----- INFORMATION ELEMENT : SHIFT
   ----0--- Shift type : locking
   -----110 New codeset ident. : local service network specific IEs
1  00100010 INFORMATION ELEMENT : SElected CALL APPearance
2  00000001 IE length : 1 octets
3  00000000 Button Nr/Call App : 0

10. CONN

08 01 81 07

PD = Q.931  CR = 0x01 Dest CONNnect  Var. = ATT_5E6

ISDN Programmer's Manual
21.4 1TR6 Primary Rate Test Scripts

These scripts simulate the user, and network sides for call setup and teardown using the 1TR6 message set over a Primary Rate link.

S2M_N.F

The S2M_N.F test script simulates the user side of a telephone call setup and teardown procedure. Functional signalling procedures are utilized and overlap sending is employed.

```
( Simulates the network side of an ISDN PCM-30 link according to overlap sending procedures as defined in the Deutsche Bundespost Recommendation 1TR6. )

" 1TR6_NSA" LOAD_MESSAGE_SET ( Select 1TR6 message set )

( Variables and buffers are defined if they have not already been. )
#IFNOTDEF CURR-BCHAN
X"0" VARIABLE CURR-BCHAN ( Contains active B Channel )
1 VARIABLE GBZ ( Contains Charging information units )
60 VARIABLE GBZ-TIME ( Charging informations unit timer value )
0 VARIABLE DATE-BUF 14 ALLOT ( Buffer for formatted Date information )
0 VARIABLE KP-BUF 20 ALLOT ( Buffer to contain KeyPad values )

( initialize CLS IE )
I#CLS ELEMENT>
  OCTET_3 INCLUDED
  #KEINE_ANGABE *COD ->SDT_STATUS !
<ELEMENT

( initialize CAUse IE )
I#CAU ELEMENT>
  OCTET_3 INCLUDED
  #REMOTE_INITIATED *COD ->CAU_VALUE !
  OCTET_4 INCLUDED
  #NATIONAL *COD ->C_CODING_STANDARD !
  #PRIVATE_NETWORK *COD ->C_LOCATION !
<ELEMENT
```
( initialize Service INDicator IE )
I#SIN ELEMENT>
   OCTET_3 INCLUDED
      #FERNSPRECHEN *COD ->SIN_SERVICE !
   OCTET_4 INCLUDED
      #ANALOG *COD ->SIN_ADD_INFO !
</ELEMENT>

: APPEND ( source string \ length \ dest string address -- )
   DUP C@ >R    ( save current length of string )
   OVER R + OVER C!    ( add lengths for new string length )
   R> + 1+    ( compute address to move source string )
   SWAP CMOVE    ( copy source string after dest string )
;

( STR+CH is used to append a single character to the end of a string. )

: STR+CH ( char \ string -- )
   DUP >R    ( save copy of string address )
   COUNT + C!    ( store character at end of string )
   1 R> C+!    ( increment string length )
;

( The following words are defined to build 1TR6 Information Elements as there )
( is no Message Builder support for this message set yet. )

: BLD_CIF ( -- ) ( Builds Charging Information IE )
I#CIF ELEMENT>
   OCTET_3 INCLUDED
      #ANZAHL_EINHEITEN *COD ->CIF_GEA !
   OCTET_4 INCLUDED
      GBZ @ <# #S #> *COD ->CIF_ANZAHL
      2DUP C!    ( store the string length )
      1+ SWAP CMOVE    ( copy the string )
</ELEMENT>
: BLD_DTE  (-- )  ( Builds Date IE )
GET_TIME ( read real time clock: ss mm hh dd mm yy )
ROT <# # # #> ( format day: "dd" )
DATE-BUF 2DUP C! 1+ SWAP CMOVE ( copy it into the date string )
SWAP <# # # 0X2E HOLD #> ( format month: ".mm" )
DATE-BUF APPEND ( append to date string )
<# # 0X2E HOLD #> ( format year: ".yy" )
DATE-BUF APPEND ( append to date string )
<# # 0X2D HOLD #> ( format hour: ":hh" )
DATE-BUF APPEND ( append to date string )
<# # 0X3A HOLD #> ( format minute: ":mm" )
DATE-BUF APPEND ( append to date string )
DROP ( drop seconds )
I#DTE ELEMENT>
OCTET_3 INCLUDED
DATE-BUF *COD ->DTE_DATA !STRING
</ELEMENT

: BLD_CHI  ( P/E -- )  ( Builds Channel Identification IE )
I#CHI ELEMENT>
ALL_EXCLUDED
OCTET_3 INCLUDED
#IMPLICIT *COD ->CID_INT_PRESENT !
?PRI *COD ->CHI_PAT1 !
( P/E from stack ) *COD ->CHI_P/E !
#NOT_D_CHANNEL *COD ->CHI_PAT2 !
#SEE_CHAN_NO *COD ->CHI_ICS !
OCTET_4 INCLUDED
#NATIONAL *COD ->CID_CODING_STANDARD !
#NUMBER *COD ->CID_NUMBER/MAP !
#B_CHANNEL_UNITS *COD ->CID_CHANNEL/MAP_TYPE !
OCTET_5 INCLUDED
CURR_BCHAN @ *COD ->CHI_CHANNEL !STRING
</ELEMENT

: BLD_DAD  (-- )  ( Builds Destination Address IE )
I#DAD ELEMENT>
OCTET_3 INCLUDED
#UNKNOWN *COD ->DAD_ADDR_TYPE !
#ISDN_PLAN *COD ->DAD_ADDR_PLAN !
OCTET_4 INCLUDED
1 COUNTER2 +! ( increment digit counter )
*COD ->DAD_ADDR 1 OVER C! ( length )
KP-BUF COUNTER2 @ + C@ SWAP 1+ C! ( digit )
</ELEMENT

IDACOM
ISDN Programmer's Manual
( The following words are defined to aid in sending messages. These words )
( use the dynamic message building method to create each message. )

: SEND_ALERT  ( -- )
  M#ALERT MESSAGE> I#CLS <SEND

: SEND_CIF    ( -- )
  BLD_CIF
  M#INFO MESSAGE> I#CIF <SEND

: SEND_CONN   ( -- )
  BLD_DTE
  M#CONN MESSAGE> I#DTE <SEND

: SEND_CONN_ACK ( -- )
  BLD_DTE
  M#CONN_ACK MESSAGE> I#DTE <SEND

: SEND_DIGIT  ( -- )
  BLD_DAD
  M#INFO MESSAGE> I#DAD <SEND

: SEND_DISC   ( -- )
  BLD_CIF BLD_DTE
  M#DISC MESSAGE> I#CAU I#CIF I#DTE <SEND

: SEND_REL    ( -- )
  BLD_CIF BLD_DTE
  M#REL MESSAGE> I#CIF I#DTE <SEND

: SEND_REL_ACK ( -- )
  M#REL_ACK MESSAGE> <SEND

: SEND_SETUP  ( -- )
  #PREFERRED BLD_CHI
  M#SETUP MESSAGE> I#CHI I#SIN <SEND

: SEND_SETUP_ACK ( -- )
  #EXCLUSIVE BLD_CHI
  M#SETUP_ACK MESSAGE> I#CHI <SEND

@endif
TCLR Clear test manager
TM_MUX_OFF Turn test manager multiplexer off
1 =CR_LENGTH Set call reference length
0X41 =CN_INT_PD Set N1 protocol discriminator

0 STATE_INIT{
  CLEAR.Keys
  " Anruf" 1 LABEL.KEY
  " TestKeys" SET_CURR_TOPIC
)STATE_INIT

0 STATE{
  UFl ?KEY ( Initiate call )
  ACTION{
    10 NEW_STATE ( Originate call )
  )ACTION

  M#SETUP ?L3_MSG ( Incoming call )
  ACTION{
    I#CHI 1 ?L3_IE ( Extract B Channel # from CHI IE )
    IF
      *DEC ->CHI_CHANNEL @ CURR-BCHAN !
      ENDIF
    #DEST =CN_CR_FLAG ( Set call reference flag )
    $MSG-CRVALUE @ =CN_CR_VALUE ( Use user's call reference )
    SEND_SETUP_ACK ( Send SETUP ACKnowledge )
    1 NEW_STATE
  )ACTION
)STATE

1 STATE_INIT{
  1 CLEAR.KEY ( Erase "Anruf" label )
  " " KP-BUF !STRING ( Initialize keypad string )
)STATE_INIT

1 STATE{
  M#INFO ?L3_MSG ( Collecting keypad digits )
  ACTION{
    I#DAD 1 ?L3_IE ( Extract digit from DAD IE )
    IF
      *DEC ->DAD_ADDR 1+ C@ KP-BUF STR+CH
      KP-BUF C@ 4 = ( 4 digits received? )
      IF
        SEND_ALERT ( Send ALERTing )
        101 30 START_TIMER ( Simulate switching delay )
        2 NEW_STATE
      ENDIF
      ENDIF
  )ACTION
)STATE

IDACOM ISDN Programmer's Manual
2 STATE{
  101 ?TIMER ( "Switching delay" timer expired )
  ACTION{
    SEND_CONN ( Send CONNECTed )
    3 NEW_STATE
  }ACTION
}STATE

3 STATE_INIT{
  " Ausloesen" 1 LABEL_KEY
  101 GBZ-TIME @ START_TIMER ( Start Charging Information timer )
  0 GBZ ! ( Initialize Charging Info. units )
}STATE_INIT

3 STATE{
  UF1 ?KEY ( Hangup )
  ACTION{
    SEND_DISC ( Send DISConnected )
    4 NEW_STATE
  }ACTION

  M#DISC ?L3_MSG ( User goes on-hook )
  ACTION{
    SEND_REL ( Send RELEASE )
    5 NEW_STATE
  }ACTION

  101 ?TIMER ( CIF timer expired )
  ACTION{
    101 GBZ-TIME @ START_TIMER ( Restart CIF timer )
    1 GBZ +! ( Increment charging units )
    SEND_CIF ( Send INFO message with CIF IE )
  }ACTION
}STATE

4 STATE{
  M#REL ?L3_MSG ( RELEASE received )
  ACTION{
    SEND_REL_ACK ( Send RELEASE ACKnowledge )
    0 NEW_STATE
  }ACTION
}STATE

5 STATE{
  M#REL_ACK ?L3_MSG ( RELEASE ACKnowledge received )
  ACTION{
    0 NEW_STATE
  }ACTION
}STATE
10 STATE_INIT{
  PROMPT " Bitte gewünschte Rufnummer eingeben (3 Ziffern) "
  prompt DUP C@ 3 = ( 3 digits entered? )
  IF
    KP-BUF !STRING ( Save Tel. No. in KP-BUF )
    0 SA DL_ESTABLISH ( Establish data link )
  ELSE
    DROP " Invalid Entry" W.ERROR ( Re-issue prompt )
  END IF
  END_PROMPT
  3 COUNTER1 ! ( # of digits to send )
  0 COUNTER2 ! ( # of digits sent )
  1 CLEAR_KEY ( Clear "Anruf" label )
}STATE_INIT

10 STATE{
  DL-EST#CONF ?L2_SERVICE ( Data link establishment confirmation )
  ACTION{
    #ORIG =CN_CR_FLAG ( Set call reference flag )
    SEND_SETUP ( Send SETUP )
    11 NEW_STATE
  }ACTION
}STATE

11 STATE{
  M#SETUP_ACK ?L3_MSG ( SETUP ACKnowledge received )
  ACTION{
    SEND_DIGIT ( Send first digit )
    12 NEW_STATE
  }ACTION
}STATE

12 STATE{
  R#RR ?RX_FRAME ( INFO frame acknowledged )
  ACTION{
    COUNTER2 @ COUNTER1 @ = ( All digits sent? )
    IF
      13 NEW_STATE
    ELSE
      SEND_DIGIT ( Send next digit )
    ENDIF
  }ACTION
}STATE

13 STATE{
  M#ALERT ?L3_MSG ( ALERTing received )
  ACTION{
    14 NEW_STATE
  }ACTION
}STATE
14 STATE{
    M#CONN ?L3_MSG   ( CONNected received )
    ACTION{
        SEND_CONN_ACK   ( Send CONNect ACKnowledge )
        3 NEW STATE
    }ACTION
}STATE
S2M_U.F

The S2M_U.F test scripts simulates the user side of a telephone call setup and teardown procedure utilizing functional signalling procedures – overlap sending. A message pool, S2M_U.P, is automatically loaded.

```plaintext
(File Title: S2M_U.F)
(Simulates the user side of and ISDN PCM-30 link according to overlap sending procedures as defined in the Deutsche Bundespost Recommendation 1TR6)

"1TR6_NSA" LOAD_MESSAGE_SET (Select 1TR6 message set)

Variables and buffers are defined if they have not already been.

#ifnotdef CURR-BCHAN
0 VARIABLE CURR-BCHAN (Contains active B Channel)
0 VARIABLE KP_BUF 20 ALLOT (Buffer to contain KeyPad values)

These words each retrieve a predefined message from the message pool and modify any internal fields before sending the message.

: SETUP_U (-- ) (Send SETUP message)
   "SETUP-U" POOL
   OVER 8+ CURR-BCHAN @ SWAP C!
   ALTER_CR SEND_I
;

: SETUP_ACK_U (-- ) (Send SETUP ACK message)
   "SETUPACK-U" POOL
   OVER 8+ CURR-BCHAN @ SWAP C!
   ALTER_CR SEND_I
;

: ALERT_U (-- ) (Send ALERTing message)
   "ALERT-U" POOL
   OVER 7 + KP_BUF 1+ SWAP 3 CMOVE
   ALTER_CR SEND_I
;
```

IDACOM

ISDN Programmer's Manual
: CONN_U ( -- ) ( Send CONNect message )
  " CONN-U" POOL
  OVER 7 + KP_BUF 1+ SWAP 3 CMOVE
  ALTER_CR SEND_I
;

( COUNTER2 is offset in KP_BUF of next char to send. )
( COUNTER1 is number of digits entered )

: DIAL_OUT ( -- ) ( Send INFO message containing KeyPad digit )
  " DIAL-U" POOL
  OVER 7 + KP_BUF 1+ COUNTER2 @ + C@ SWAP C!
  1 COUNTER2 +!
  ALTER_CR SEND_I
;

+KP ( char -- ) ( Append the next digit to the keypad string )
  KP_BUF C@ 1+
  DUP KP_BUF C!
  KP_BUF + C!
;

#ENDIF

TCLR
  " S2M_U.P" LOAD_MESSAGES
  TM_MUX_OFF
  1 =CR_LENGTH

0 STATE_INIT{
    CLEAR KEYS
    " Anruf" 1 LABEL_KEY
    " TestKeys" SET_CURR_TOPIC
}STATE_INIT
0 STATE{
  UF1 ?KEY
  ACTION{
    10 NEW_STATE
  )ACTION

  M#SETUP ?L3_MSG
  ACTION{
    I#CHI 1 ?L3_IE
    IF
      *DEC ->CHI_CHANNEL @ CURR-BCHAN !
    ENDIF
    DEST_SIDE
    $MSG-CRVALUE @ RX-CALLREF !
    SETUP_ACK_U
    1 NEW_STATE
  )ACTION
)STATE

1 STATE_INIT{
  1 CLEAR_KEY
  " " KP_BUF !STRING
)STATE_INIT

1 STATE{
  M#INFO ?L3_MSG
  ACTION{
    I#DAD 1 ?L3_IE
    IF
      *DEC ->DAD_ADDR 1+ C@ +KP
      KP_BUF C@ 3 =
      IF
        ALERT_U
        101 30 START_TIMER
        3 NEW_STATE
      ENDIF
    ENDIF
  )ACTION
)STATE

3 STATE{
  101 ?TIMER
  ACTION{
    CONN_U
    4 NEW_STATE
  )ACTION
)STATE
4 STATE[
    M#CONN_ACK ?L3_MSG ( CONNect ACKnowledge received )
    ACTION[
        5 NEW_STATE
    ]ACTION
]STATE

5 STATE_INIT[
    " Ausloesen" 1 LABEL_KEY
]STATE_INIT

5 STATE[
    UF1 ?KEY ( Hang up )
    ACTION[
        " DISC-U" POOL ALTER_CR SEND_I ( Send DISConnect )
        6 NEW_STATE
    ]ACTION

    M#DISC ?L3_MSG ( Far end disconnects )
    ACTION[
        " REL-U" POOL ALTER_CR SEND_I ( Send RELease )
        7 NEW_STATE
    ]ACTION
]STATE

6 STATE[
    M#REL ?L3_MSG ( RELease received )
    ACTION[
        " REL_ACK" POOL ALTER_CR SEND_I ( Send RELease ACKnowledge )
        0 NEW_STATE
    ]ACTION
]STATE

7 STATE[
    M#REL_ACK ?L3_MSG ( RELease ACKnowledge received )
    ACTION[
        0 NEW_STATE
    ]ACTION
]STATE
10 STATE_INIT{
    PROMPT "Bitte gewünschte Rufnummer eingeben (4 Ziffern)"
    prompt DUP C@ 4 = (4 digits entered?)
    IF
        KP_BUF !STRING (Save Tel. No. in KP_BUF)
        0 SA DL_ESTABLISH (Establish data link)
    ELSE
        DROP "Invalid Entry" W.ERROR
        10 NEW_STATE (Re-issue prompt)
    ENDIF
    END_PROMPT
    4 COUNTER1 ! ( # of digits to send)
    0 COUNTER2 ! ( # of digits sent)
    1 CLEAR_KEY (Erase "Anruf" label)
}STATE_INIT

10 STATE{
    DL-EST#CONF ?L2_SERVICE (Data link establishment confirmation)
    ACTION{
        ORIG_SIDE (Set the call reference flag)
        SETUP_U (Send SETUP)
        11 NEW_STATE
    }ACTION
}STATE

11 STATE{
    M#SETUP_ACK ?L3_MSG (SETUP ACKnowledgement received)
    ACTION{
        DIAL_OUT (Send the first digit)
        12 NEW_STATE
    }ACTION
}STATE

12 STATE{
    R#RR ?RX_FRAME (INFO frame acknowledgement received?)
    ACTION{
        COUNTER2 @ COUNTER1 @ = (All digits sent?)
        IF
            13 NEW_STATE
        ELSE
            DIAL_OUT (Send the next digit)
        ENDIF
    }ACTION
}STATE

13 STATE{
    M#ALERT ?L3_MSG (ALERTing received)
    ACTION{
        14 NEW_STATE
    }ACTION
}STATE
Message Pool Descriptions

There are eight pool entries in the S2M_U.P message pool file as described in the following list. Each pool entry is shown in hexadecimal and complete formats.

1. SETUP-U

41 01 00 05

18 03 A9 C3 0D
2B 08 1B 01 49 44 41 43 4F 4D
6C 04 80 32 30 38
9E
01 02 01 01
9F
01 03 A5 00 90
9F
03 0C 30 00 00 80 0E 00 00 80 01 00 00 8D

PD = N1 CR = 0x00 Orig SETUP Var. = lTR6_NSA
1 00011000 INFORMATION ELEMENT : Channel Identification
2 00000011 IE length : 3 octets
3 1------- Extension bit : not continued
-0------ Interface ident. : implicitly identified
--1----- Interface type : primary interface
-----0---- Spare :
------1 P/E : vorgeschriebener Kanal
-------0-- D-Channel Indicator : not D-channel
--------0 ICS : siehe Channel Number
4 1------- Extension bit : not continued
-10----- Coding standard : national standard
-----0---- Number/Map : number
-----0011 Chan./Map type : B-channel units
5 ****** Channel number : OD
1 00101000 INFORMATION ELEMENT : DiPlay
2 00001000 IE length : 8 octets
3 ****** Displayinhalt : [00][00]IDACOM]
1 01101100 INFORMATION ELEMENT : Origination Address
2 00000100 IE length : 4 octets
3 1------- Extension bit : not continued
-000---- Type of address : unknown
------0000 Numbering plan : unknown
4 ****** Address digits : [202]
1 1001---- INFORMATION ELEMENT : SHIf
### 2. DIAL-U

```
41 01 00 6D
70 02 80 31
```

<table>
<thead>
<tr>
<th>PD = N1</th>
<th>CR = 0x00</th>
<th>Orig INFORMATION</th>
<th>Var. = 1TR6_NSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01110000</td>
<td>INFORMATION ELEMENT: Destination Address</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>00000010</td>
<td>IE length: 2 octets</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0--------</td>
<td>Extension bit: not continued</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-000-----</td>
<td>Type of address: unknown</td>
<td></td>
</tr>
<tr>
<td></td>
<td>----000</td>
<td>Numbering plan: unknown</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>*********</td>
<td>Address digits: [1]</td>
<td></td>
</tr>
</tbody>
</table>

### 3. REL-U

```
41 01 00 4D
9E
03 00
```

<table>
<thead>
<tr>
<th>PD = N1</th>
<th>CR = 0x00</th>
<th>Orig RELEASE</th>
<th>Var. = 1TR6_NSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1001------</td>
<td>INFORMATION ELEMENT: SHIFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-----1---</td>
<td>Shift type: Codeumschaltung ohne Feststellung</td>
<td></td>
</tr>
<tr>
<td></td>
<td>------110</td>
<td>Codeset ident.: Codesatz 6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>00000011</td>
<td>INFORMATION ELEMENT: DaFE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>00000000</td>
<td>IE length: 0 octets</td>
<td></td>
</tr>
</tbody>
</table>

### 4. REL_ACK

```
41 01 80 5A
```

<table>
<thead>
<tr>
<th>PD = N1</th>
<th>CR = 0x00</th>
<th>Dest RELEASE ACKnowledge</th>
<th>Var. = 1TR6_NSA</th>
</tr>
</thead>
</table>
5. DISC-U

41 01 00 45

9E
03 00
9F
01 01 81
9F
03 0C 00 00 80 00 00 80 00 00 80

PD = N1  CR = 0X00  Orig  DISConnect  Var. = 1TR6_NSA

1 1001---- INFORMATION ELEMENT : Shift
    -----1---- Shift type : Codeumschaltung ohne Feststellung
    -----110 Codeset ident. : Codesatz 6

1 00000011 INFORMATION ELEMENT : DaTE

2 00000000 IE length : 0 octets

1 1001---- INFORMATION ELEMENT : Shift
    -----1---- Shift type : Codeumschaltung ohne Feststellung
    -----111 Codeset ident. : Codesatz 7

1 00000001 INFORMATION ELEMENT : Set(07)/Code(01)

2 00000001 IE length : 1 octets

3 ******** IE contents : 81

1 1001---- INFORMATION ELEMENT : Shift
    -----1---- Shift type : Codeumschaltung ohne Feststellung
    -----111 Codeset ident. : Codesatz 7

1 00000011 INFORMATION ELEMENT : Set(07)/Code(03)

2 00001100 IE length : 12 octets

3 ******** IE contents : 00 00 00 80 00 00 80 00 00 80

6. SETUPACK-U

PD = N1  CR = 0X101  Dest  SETUP  ACKnowledge  Var. = 1TR6_NSA

1 00011000 INFORMATION ELEMENT : Channel Identification

2 00000011 IE length : 3 octets

3 1------- Extension bit : not continued
   -0------ Interface ident. : implicitly identified
   --1----- Interface type : primary interface
   ----0--- Spare :
   ----0--- P/E : bevorzugter Kanal
   ----0--- D-Channel Indicator : not D-channel
   -----01 ICS : siehe Channel Number

4 1------- Extension bit : not continued
   -10----- Coding standard : national standard
   ----0---- Number/Map : number
   ----0011 Chan./Map type : B-channel units

5 ******** Channel number : 1E
7. ALERT-U

41 01 81 01

0C 04 81 32 30 32
28 08 1B 01 49 44 41 43 4F 4D
9E
07 00
9F
01 03 A5 10 90
9F
03 0C 30 00 00 80 0E 00 00 80 00 00 80 8D

PD = N1    CR = 0x01    Dest ALERTing
1 00001100 INFORMATION ELEMENT : Connected Address
2 00000100 IE length : 4 octets
3 1-------- Extension bit : not continued
-000---- Type of address : unknown
-----0001 Numbering plan : ISDN numbering plan Rec. E.164
4 ******** Address digits : [202]
1 00101000 INFORMATION ELEMENT : Display
2 00001000 IE length : 8 octets
3 ******** Displayinhalt : [<00><00>IDACOM]
1 1001---- INFORMATION ELEMENT : Shift
-----1--- Shift type : Codeumschaltung ohne Feststellung
-----110 Codeset ident. : Codesatz 6
1 00000111 INFORMATION ELEMENT : Status des geruf Tln
2 00000000 IE length : 0 octets
1 1001---- INFORMATION ELEMENT : Shift
-----1--- Shift type : Codeumschaltung ohne Feststellung
-----111 Codeset ident. : Codesatz 7
1 00000011 INFORMATION ELEMENT : Set(07)/Code(01)
2 00000011 IE length : 3 octets
3 ****** IE contents : A5 10 90
1 1001---- INFORMATION ELEMENT : Shift
-----1--- Shift type : Codeumschaltung ohne Feststellung
-----111 Codeset ident. : Codesatz 7
1 00000011 INFORMATION ELEMENT : Set(07)/Code(03)
2 00001100 IE length : 12 octets
3 ****** IE contents : 30 00 00 80 0E 00 00 80 00 00 00 8D
## 8. CONN–U

```
41 01 81 07
0C 04 81 32 30 32
28 08 1B 01 49 44 41 43 4F 4D
9E
03 00
9F
01 03 A5 10 90
9F
03 0C 30 00 00 80 0E 00 00 80 00 00 8D
```

<table>
<thead>
<tr>
<th>CR</th>
<th>IE length</th>
<th>IDE contents</th>
<th>Codeset ident.</th>
<th>Shift type</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001100</td>
<td>4 octets</td>
<td>Connected Address</td>
<td>Codesatz 6</td>
<td></td>
</tr>
<tr>
<td>00001000</td>
<td>8 octets</td>
<td>Displayinhalt: [&lt;00&gt;&lt;00&gt;IDAcom]</td>
<td>Codesatz 7</td>
<td></td>
</tr>
<tr>
<td>00000001</td>
<td>3 octets</td>
<td>Set(07)/Code(01)</td>
<td>Codesatz 7</td>
<td></td>
</tr>
<tr>
<td>00000001</td>
<td>3 octets</td>
<td>A5 10 90</td>
<td>Codesatz 7</td>
<td></td>
</tr>
<tr>
<td>00001100</td>
<td>12 octets</td>
<td>30 00 00 80 0E 00 00 80 00 00 8D</td>
<td>Codesatz 7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PD</th>
<th>CR</th>
<th>Dest</th>
<th>Connnect</th>
<th>Var.</th>
<th>ISDN numbering plan Rec. E.164</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>0X01</td>
<td>CONNect</td>
<td>V</td>
<td>1TR6_NSA</td>
<td></td>
</tr>
</tbody>
</table>

---

**PD = N1**  
**CR = 0X01**  
**Dest = CONNect**  
**Var. = 1TR6_NSA**  

**IE length**  
**Extension bit**  
**Type of address**  
**Numbering plan**  
**Address digits**  
**Displayinhalt**  
**Shift type**  
**Codeset ident.**  
**IE contents**  

---

**ISDN Programmer's Manual**: IDACOM
21.5 Layer 3 Simulation

This test script is composed of three files. The file L3SIM.F is loaded first. When L3SIM.F has finished loading, it automatically loads L3S_USER.F or L3S_NTWK.F, depending on whether the application has been configured for user or network emulation.

L3SIM.F

" CCITT_1988" LOAD_MESSAGE_SET ( make sure that ccitt message set is loaded )
L3_OFF ( Turn off L3 Emulation )
TM_MUX_ON ( Turn Test Manager Multiplexer on )

( Define these words only if they have not already been defined. )
#IFNOTDEF BEEP-ALTER

( These words are used to simulate a phone ring via the PT's speaker. )

0 VARIABLE BEEP-ALTER

: RING_ON
   BEEP-ALTER @
   IF
      #USER1 10 =CN_TIMER_DUR
      OFF BEEP-ALTER !
   ELSE
      #USER1 20 =CN_TIMER_DUR
      ON BEEP-ALTER !
   ENDIF
   #USER1 CN_START_TIMER
   220 BEEP_TONE !
   4 BEEP_DUR !
   BEEP BEEP
;

: RING_OFF
   #USER1 CN_STOP_TIMER
   OFF BEEP-ALTER !
   82 BEEP_TONE !
   2 BEEP_DUR !
;
The following words are used to dynamically build Information Elements
used in this simulation. Most of the parameters are given default values;
some are passed via the stack.

: SETUP_BEARER_CAP  ( transfer mode -- )
  I#BEARER_CAP ELEMENT>
    ALL_EXCLUDED
    OCTET_3 INCLUDED
      #CCITT
      #UNRESTRICTED
      OCTET_4 INCLUDED
        ( transfer mode )
        #64KBIT/S
    OCTET_4A INCLUDED
      #DEFAULT
      #POINT_TO_POINT
      #DEMAND
      OCTET_4B INCLUDED
        #BIDIRECT_Symmetric
        #64KBIT/S

<ELEMENT
;

: SETUP_CALLED_NUM  ( number -- )
  I#CALLED_NUM ELEMENT>
    ALL_INCLUDED
      #LOCAL_DIRECTORY
      #NATIONAL_PLAN
      ( number )

<ELEMENT
;

: SETUP_CALL_STATE  ( state -- )
  I#CALL_STATE ELEMENT>
    ALL_INCLUDED
      #CCITT
      ( state )

<ELEMENT
;
These words are defined to simplify construction of the test script:

( This word returns true if a valid L.3 message was received and the call reference field applies to this connection. )

\begin{verbatim}
: ?VALID_MSG   ( Msg-id --- boolean )
  \text{?L3_VALID_MSG DUP ( Valid Layer 3 Signalling message ? )}
  IF
  \text{$MSG-CRLEN @ ?CR_LENGTH = AND ( Call Reference matched ? )}
  \text{$MSG-CRVALUE @ ?CN_CR_VALUE = AND}
  \text{$MSG-CRFLAG @ 0= ?CN_CR_FLAG = AND}
  ENDIF
\end{verbatim}
( This word attempts to find an unused B Channel for use by the call. The )
( preferred channel is attempted first. )

: NEGOTIATE_CHANNEL    ( --- Channel # \ YES or NO )
   *DEC ->CID_INT_TYPE @ #OTHER_INTERFACE =   ( Not Basic interface? )
   *DEC ->BC_TRANSFER_MODE @ #CIRCUIT_MODE = AND   ( Circuit mode? )
   *DEC ->CID_DCHANNEL @ #NOT_D_CHANNEL = AND   ( B Channel wanted? )
   *DEC ->CID_INFO_CHAN_SEL @ #AS_INDICATED = AND   ( Indicated in CID IE? )
   *DEC ->CID_NUMBER/MAP @ #NUMBER = AND   ( Number used? )
   *DEC ->CID_CHANNEL/MAP_TYPE @ #B_CHANNEL_UNITS = AND

   IF
   *DEC ->CID_NUMBER @
   DUP ?BC_ALLOC 0=   ( Is selected B Channel available? )
   IF
   YES
   ELSE
   DROP
   *DEC ->CID_PREF/EXCL @ #EXCLUSIVE =   ( only use indicated? )
   IF
   NO   ( The selected channel is not available )
   ELSE
   ?BC_SELECT
   ?DUP   ( Any B Channels available? )
   ENDIF
   ENDIF
   ELSE
   NO
   ENDIF

;
This word is used to indicate, via the CAUSE IE, a message error.

: L3_ERROR ( --- )

$MSG-ERROR @

IF

#INVALID_MESSAGE_UNSPEC
ELSE

M#UNDEF ?L3_MSG
IF

#MESSAGE_TYPE_UNIMPL
ELSE

$MSG-ID @ ?L3_MAND_IE 0=
IF

#MAND_IE_MISSING
ELSE

$MSG-ID @ ?L3_UNEXP_IE
?L3_IE_ORDER 0= OR
IF

#INVALID_MESSAGE_UNSPEC
ELSE

#MESSAGE_UNDEFINED
ENDIF
ENDIF
ENDIF
ENDIF
SETUP_CAUSE

( This word reports the status using trace statements. )

: REPORT_PEER_STATUS

T." ***** Received STATUS message ***** " TCR
T." Peer entity state : " *DEC ->CS_CALL_STATE @ T. TCR
T." Cause class : " *DEC ->C_CAUSE_VALUE @ T. TCR

;#ENDIF

( Load the appropriate Test Manager state machine: )

#IF USR*NET @ NT = #IS_TRUE
" L3S_NTWK.F" EXEC
#ELSE
" L3S_USER.F" EXEC
#ENDIF
A reliable data link must be first established. The interface structure is basic access en-bloc sending and circuit switching TE: Type 1 terminal, i.e. one connection per data link.

---

File Title: L3S_NTWK.F

This script simulates the network in a point to multi-point configuration.

---

Two counter variables are used in this script:
- COUNTER1: retransmission counter for T303
- COUNTER2: Release Complete received before T303 expiry

TCLR (Initialize test manager)

MAKE TM_INIT CLEAR_ALL_CNS;

STATE_INIT[
  0 COUNTER1 ! (Initialize retransmission counter)
  NO COUNTER2 ! (No RELEASE COMPLETE received yet)
  CLEAR_KEYS
  " MAKE CALL" 1 LABEL_KEY
  " TestKeys" SET_CURR_TOPIC
]STATE_INIT
0 STATE
  UFI ?KEY
  ACTION
    DL_ESTABLISH ( Incoming call to user )
    ?CNALLOC ( Establish the Link )
    IF
      =CN ( Set the current connection controller )
      #ORIG 0 CN_INIT ( Initialize CN )
      ?BC_SELECT ?DUP ( Allocate an idle B-Channel )
      IF
        DUP =CN_CHANNEL_NUM ( Set the channel id )
        BC_ALLOC ( Reserve the channel )
        SELECT_CR =CN_CR_VALUE ( Select Call reference value )
        0 CES* @ =CN_CEI ( Set Connection Endpoint Identifier )
        #CIRCUIT_MODE SETUP_BEARER_CAP
        ?CN_CHANNEL_NUM #EXCLUSIVE SETUP_CHANNEL_ID
        "5416320" SETUP_CALLED_NUM
        #ALERTING_ON_0 SETUP_SIGNAL
        T." ----> Send SETUP message * TCR
        M#SETUP MESSAGE>
        I#BEARER_CAP I#CHANNEL_ID I#SIGNAL I#CALLED_NUM
        <SEND
        #T303 #T303_DUR =CN_TIMER_DUR
        #T303 CN_START_TIMER ( Start timer T303 )
        6 NEW_STATE
      ELSE ( No channel available )
        CN_DEALLOC ( Clear connection controller )
        " NO CHANNEL AVAILABLE " W.ERROR
        ENDIF
      ELSE ( Network busy )
        " NO CONNECTION MODULE AVAILABLE " W.ERROR
        ENDIF
    ENDIF
  )ACTION
  M#SETUP ?L3_VALID_MSG
  ACTION
    I#CHANNEL_ID ?L3_IECOUNT 0>
    I#CALLED_NUM ?L3_IECOUNT 0> AND ( All required info existed ? )
    IF
      ?CNALLOC ( Allocate connection controller )
      IF
        =CN ( Set current connection controller )
        #DEST 0 CN_INIT ( Initialize the controller )
        NEGOTIATE_CHANNEL
  ENDIF
IF

MSAPI* @ CES* @ =CN_CEI
DUP =CN_CHANNEL_NUM    ( Save the channel id )
BC_ALLOC             ( Reserve the channel )
?CN_CHANNEL_NUM #EXCLUSIVE SETUP_CHANNEL_ID
T. " ----> Send CALL PROCEEDING message " TCR
M#CALL_PROC MESSAGE>
       I#CHANNEL_ID
<SEND
3 NEW_STATE
ELSE    ( No B-Channel available )
   #NO_CHANNEL_AVAIL SETUP_CAUSE
T. " ----> Send RELEASE COMPLETE message " TCR
M#REL_COM MESSAGE>
       I#CAUSE
<SEND
CN_DEALLOC
   ( Clear connection controller )
ENDIF
ELSE    ( No connection controller )
   #SWITCH_CONGESTION SETUP_CAUSE
T. " ----> Send RELEASE COMPLETE message " TCR
M#REL_COM MESSAGE>
       I#CAUSE
<SEND
ENDIF
ELSE    ( Insufficient Information, Overlap Sending not supported )
   #SERVICE_UNIMPL_UNSPEC SETUP_CAUSE
T. " ----> Send RELEASE COMPLETE message " TCR
M#REL_COM MESSAGE>
       I#CAUSE
<SEND
ENDIF
)ACTION

M#STATUS ?L3_VALID_MSG
ACTION{
      REPORT_PEER_STATUS
}ACTION
)STATE

3 STATE_INIT{
   CLEAR_KEYS
   " CONNECT" 2 LABEL_KEY
   " REJECT" 3 LABEL_KEY
)STATE_INIT
3 STATE{ ( OUTGOING CALL PROCEEDING )
  UF2 ?KEY ( Call is accepted and connected )
  ACTION{
    ?CN_CE1 SA DROP
    ?CN_CHANNEL_NUM BC_CONN ( Connected to the B channel )
    T." ----> Send CONNECT message " TCR
    M#CONN MESSAGE>
    <SEND
    10 NEW_STATE
  }ACTION

  M#STATUS ?L3_VALID_MSG ( Received STATUS message or )
  UF3 ?KEY OR ( Call is rejected -> initiate clearing )
  ACTION{
    #USER_BUSY SETUP_CAUSE
    T." ----> Send DISCONNECT message " TCR
    M#DISC MESSAGE>
    I#CAUSE
    <SEND
    #T305 #T305_DUR =CN_TIMER_DUR
    #T305 CN_START_TIMER ( Start timer T305 )
    12 NEW_STATE
  }ACTION

  M#DISC ?L3_VALID_MSG ( Calling user releases the call )
  ACTION{
    #USER_BUSY SETUP_CAUSE
    T." ----> Send RELEASE message " TCR
    M#REL MESSAGE>
    I#CAUSE
    <SEND
    #T308 #T308_DUR =CN_TIMER_DUR
    #T308 CN_START_TIMER ( Start timer T308 )
    19 NEW_STATE
  }ACTION
}

6 STATE{ ( CALL PRESENT )
  M#CONN ?L3_VALID_MSG ( Received a CONNECT message )
  ACTION{
    #T303 CN_STOP_TIMER ( Stop timer T303 )
    0 COUNTER1 !
    8 NEW_STATE
  }ACTION

  M#REL_COM ?L3_VALID_MSG
  ACTION{
    YES COUNTER2 ! ( Received RELEASE COMPLETE message )
    ( Received REL COM )
  }ACTION
M#STATUS ?L3_VALID_MSG
ACTION{
    REPORT_PEER_STATUS
    ?CN_CHANNEL_NUM BC_FREE ( Release B channel )
    CLEAR_CR
    ( Clear call reference & connection module )
    0 NEW_STATE
}ACTION

#T303 ?CN_TIMER ( Timer T303 expired )
ACTION{
    COUNTER1 @ 0= COUNTER2 @ NO = AND
    IF
        1 COUNTER1 +! ( Increment the counter )
        T. " ----> Re-send SETUP message " TCR
        M#SETUP MESSAGE>
        I#BEARER_CAP I#CHANNEL_ID I#SIGNAL I#CALLED_NUM
        <SEND ( Retransmit the SETUP message )
        #T303 #T303_DUR =CN_TIMER_DUR
        #T303 CN_START_TIMER ( Start timer T303 )
    ELSE
        ?CN_CHANNEL_NUM BC_FREE ( Release B channel )
        CLEAR_CR
        ( Clear call reference & connection module )
        0 NEW_STATE
    ENDIF
}ACTION

?RX_DATA ( Check received message error )
ACTION{
    L3_ERROR ( Check error )
    T. " ----> Send DISCONNECT message " TCR
    M#DISC MESSAGE>
    I#CAUSE
    <SEND
    #T305 #T305_DUR =CN_TIMER_DUR
    #T305 CN_START_TIMER
    12 NEW_STATE
}ACTION
}STATE

8 STATE_INIT{
    CLEAR_KEYS
    " CONNECT" 2 LABEL_KEY
    " DISCONNECT" 3 LABEL_KEY
}STATE_INIT
8 STATE{
  ( NETWORK CONNECT )
  UF2 ?KEY ACTION{
    ( Accept the call )
    #ALERTING_OFF SETUP_SIGNAL
    T." ----> Send CONNECT ACKNOWLEDGE message " TCR
    M#CONN_ACK MESSAGE>
    I#SIGNAL
    <SEND
    ?CN_CHANNEL_NUM BC_CONN   ( Connect B channel )
    10 NEW_STATE
  }ACTION

  UF3 ?KEY ACTION{
    ( Reject the call )
    #NORMAL_CLEARING SETUP_CAUSE
    T." ----> Send DISCONNECT message " TCR
    M#DISC MESSAGE>
    I#CAUSE
    <SEND
    #T305 #T305_DUR =CN_TIMER_DUR
    #T305 CN_START_TIMER     ( Start timer T305 )
    12 NEW_STATE
  }ACTION

  M#DISC ?L3_VALID_MSG
  ACTION{
    #NORMAL_CLEARING SETUP_CAUSE
    T." ----> Send RELEASE message " TCR
    M#REL MESSAGE>
    I#CAUSE
    <SEND
    19 NEW_STATE
  }ACTION
}

10 STATE_INIT{
  CLEAR KEYS
  " DISCONNECT" 2 LABEL_KEY
}

10 STATE{
  ( ACTIVE )
  UF2 ?KEY
  ( Network initiates the release )
  ACTION{
    ?CN_CHANNEL_NUM BC_DISC   ( Disconnect channel-in-use )
    #NORMAL_CLEARING SETUP_CAUSE
    T." ----> Send DISCONNECT message " TCR
    M#DISC MESSAGE>
    I#CAUSE
    <SEND
    #T305 #T305_DUR =CN_TIMER_DUR
    #T305 CN_START_TIMER       ( Start timer T305 )
    12 NEW_STATE
  }ACTION
M#DISC ?L3_VALID_MSG
ACTION{
    ( User wants to release the call )
?CN_CHANNEL_NUM BC_DISC  
    ( Disconnect the channel )
*DEC ->C_CAUSE_VALUE @ SETUP_CAUSE
T." ----> Send RELEASE message " TCR
M#REL MESSAGE>
    I#CAUSE
<SEND
#T308 #T308_DUR =CN_TIMER_DUR
#T308 CN_START_TIMER  
    ( Start timer T308 )
19 NEW_STATE
}ACTION
}STATE

12 STATE_INIT{
    CLEAR_KEYS
}STATE_INIT

12 STATE{
    ( DISCONNECT IND )
M#REL ?L3_VALID_MSG
ACTION{
    ?CN_CHANNEL_NUM BC_FREE  
    ( Free the channel )
*DEC ->C_CAUSE_VALUE @ SETUP_CAUSE
T." ----> Send RELEASE COMPLETE message " TCR
M#REL_COM MESSAGE>
    I#CAUSE
<SEND
    CLEAR_CR  
    ( Clear Call reference )
0 NEW_STATE
}ACTION

#T305 ?CN_TIMER  
    ( Timer T305 expired )
ACTION{
    #NORMAL_CLEARING SETUP_CAUSE
T." ----> Send RELEASE message " TCR
M#REL MESSAGE>
    I#CAUSE
<SEND
#T308 #T308_DUR =CN_TIMER_DUR
#T308 CN_START_TIMER  
    ( Start timer T308 )
19 NEW_STATE
}ACTION
}STATE

19 STATE_INIT{
    CLEAR_KEYS
}STATE_INIT
19 STATE{
  M#REL_COM ?L3_VALID_MSG          ( Received RELEASE COMPLETE message )
  ACTION{
    ?CN_CHANNEL_NUM BC_FREE        ( Release B Channel )
    CLEAR_CR                      ( Release call reference )
    0 NEW_STATE
  }ACTION

#T308 ?CN_TIMER                   ( Timer T308 expired )
ACTION{
  COUNTER1 @ 2 <
  IF
    1 COUNTER1 +!                    ( Increment counter )
    #NORMAL_CLEARING SETUP_CAUSE
    T. " ----> Re-send RELEASE message " TCR
    M#REL MESSAGE>
    I#CAUSE
    <SEND
    #T308 #T308_DUR =CN_TIMER_DUR
    #T308 CN_START_TIMER            ( Restart timer T308 )
  ELSE
    ?CN_CHANNEL_NUM BC_FREE        ( Release B Channel )
    CLEAR_CR                      ( Release call reference )
    0 NEW_STATE
  ENDIF
}ACTION
}STATE
L3S_USER.F

A reliable data link must first be established. The interface structure is basic access en-bloc sending and circuit switching TE: Type 1 terminal, i.e. one connection per data link.

( File Title: L3S_USER.F )
( This script simulates the user side in a point to point configuration )
( ----------------------------------------------- )

TCLR ( Initialize test manager )

MAKE TM_INIT CLEAR_ALL_CNS ;

( Three counter variables are used as flags that may be changed to alter the operation of the simulation: )

YES COUNTER1 ! ( true if the terminal is compatible )
YES COUNTER2 ! ( true if the terminal accepts incoming calls )
YES COUNTER3 ! ( true if the terminal sends CONNect ACKnowledge in )

0 STATE_INIT{
   CLEAR_KEYS ( Erase TestKeys function key labels )
   " MAKE CALL" 1 LABEL_KEY ( Label FK #1 )
   " TestKeys" SET_CURR_TOPIC ( Switch to the TestKeys topic )
}

STATE_INIT
0 STATE{
  ( Initial State )
  UFl ?KEY
  ( User initiates call )
  ACTION{
    DL_ESTABLISH
    ( Establish the link )
    ?CN_ALLOC
    ( Allocate connection control module )
    IF
    =CN
    ( Set the current CN )
    #ORIG 0 CN_INIT
    ( Initialize CN )
    SELECT_CR =CN_CR_VALUE
    ( Select Call reference value )
    0 CES* @ =CN_CEI
    ( Set Connection Endpoint Identifier )
    #CIRCUIT_MODE
    SETUP_BEARER_CAP
    #B1_CHANNEL #PREFERRED SETUP_CHANNEL_ID
    " 5416320" SETUP_CALLED_NUM
    T." ----> Send SETUP message " TCR
    M#SETUP MESSAGE>
    I#BEARER_CAP I#CHANNEL_ID I#CALLED_NUM
    <SEND
    1 NEW_STATE
    ELSE
    " NO CONNECTION MODULE AVAILABLE " W.ERROR
    ENDIF
  }ACTION

M#SETUP ?L3_VALID_MSG
  ( Incoming Call )
  ACTION{
    I#CALLED_NUM ?L3_IECOUNT 0
    ( Called num IE present ? )
    IF
    COUNTER1 @
    ( Compatible terminal ? )
    IF
    ?CN_ALLOC
    ( Allocate connection control module )
    IF
    =CN
    ( Set the current connection handler )
    #DEST 0 CN_INIT
    ( Initialize the connection )
    NEGOTIATE_CHANNEL
    ( Channel available ? )
    IF
    DUP =CN_CHANNEL_NUM
    ( Set the allocate channel )
    BC_ALLOC
    ( Reserved the channel )
    COUNTER2 @
    IF
    I#SIGNAL ?L3_IECOUNT
    ( Signal IE present ? )
    IF
    RING_ON
    ( Turn ring on )
    ENDIF
    ( Turn ring on )
    MSAPI* @ CES* @ =CN_CEI
    ( Save CEI value )
    T." ----> Send CONNECT message " TCR
    M#CONN MESSAGE> <SEND
    ( Send CONNECT message )
    8 NEW_STATE

IDACOM
ISDN Programmer's Manual
ELSE
  #NORMAL_CLEARING SETUP_CAUSE
  T.‘ ———> Send RELEASE COMPLETE message ' TCR
  M#REL_COM MESSAGE> I#CAUSE <SEND
  ?CN_CHANNEL_NUM BC_FREE ( Free reserved B channel )
  CN_DEALLOCC ( Reset connection handler )
ENDIF
ELSE ( Channel not available )
  #NO_CHANNEL_AVAIL SETUP_CAUSE
  T.‘ ———> Send RELEASE COMPLETE message ' TCR
  M#REL_COM MESSAGE> I#CAUSE <SEND
  CN_DEALLOCC ( Reset connection handler )
ENDIF
ELSE ( User busy )
  1 ( #NORMAL1 ) #USER_BUSY SYS_Q931_CAUSE
  T.‘ ———> Send RELEASE COMPLETE message ' TCR
  M#REL_COM SM_MSG
ENDIF
ELSE ( Incompatible terminal )
  #CALL_REJECTED SETUP_CAUSE
  T.‘ ———> Send RELEASE COMPLETE message ' TCR
  M#REL_COM MESSAGE> I#CAUSE <SEND
ENDIF
ELSE ( Overlap sending not supported )
  #SERVICE_UNIMPL_UNSPEC SETUP_CAUSE
  T.‘ ———> Send RELEASE COMPLETE message ' TCR
  M#REL_COM MESSAGE> I#CAUSE <SEND
ENDIF
}ACTION

M#STATUS ?L3_VALID_MSG ( Network requests status )
ACTION[
  REPORT_PEER_STATUS
]ACTION

}STATE

1 STATE_INIT[
  " DISCONNECT" 2 LABEL_KEY
]STATE_INIT

1 STATE[
  ( Call Initialization )
  ( Clear the call )
  UF2 ?KEY
  ACTION[
    #NORMAL_CLEARING SETUP_CAUSE
    T.‘ ———> Send DISCONNECT message ' TCR
    M#DISC MESSAGE> I#CAUSE <SEND
    11 NEW_STATE
  ]ACTION
M#CALL_PROC ?VALID_MSG (Received a CALL PROC message)
ACTION{
  *DEC ->CID_NUMBER @ DUP (Selected Channel)
  ?BC_ALLOC 0= (Channel available?)
  IF
    DUP =CN_CHANNEL_NUM (Save the channel number)
    BC_ALLOC (Allocate the reserved channel)
    3 NEW_STATE
  ELSE
    DROP (Channel not available)
    #NO_CHANNEL_AVAIL SETUP_CAUSE
    T."----> Send DISCONNECT message " TCR
    M#DISC MESSAGE> I#CAUSE <SEND
    11 NEW_STATE
ENDIF
)ACTION

M#REL_COM ?L3_VALID_MSG (Received a RELEASE COMPLETE message)
ACTION{
  CLEAR_CR
  0 NEW_STATE
)ACTION

M#STATUS ?L3_VALID_MSG (Status Request, treat as RELEASE COMPLETE)
ACTION{
  REPORT_PEER_STATUS
  CLEAR_CR
  0 NEW_STATE
)ACTION

?RX_DATA (Check received message error)
ACTION{
  L3_ERROR (Check error)
  #CALL_INIT SETUP_CALL_STATE
  T."----> Send STATUS message " TCR
  M#STATUS SM_MSG
  CLEAR_CR (Clear call reference & connection handler)
  0 NEW_STATE
)ACTION
}STATE

3 STATE_INIT{
  CLEAR_KEYS
}STATE_INIT
3 STATE { (Outgoing call proceeding) 
M#ALERT ?L3_VALID_MSG (Calling user has been alerted) 
ACTION{ 
4 NEW_STATE 
}ACTION 
M#CONN ?L3_VALID_MSG (Call is connected) 
ACTION{ 
?CN_CHANNEL_NUM BC_CONN (Channel is connected) 
COUNTER3 @ (Option to send CONNECT ACK) 
IF 
T." ----> Send CONNECT ACKNOWLEDGE message " TCR 
M#CONN_ACK MESSAGE> <SEND 
ENDIF 
10 NEW_STATE 
}ACTION 

M#DISC ?L3_VALID_MSG (Call is rejected) 
ACTION{ 
?CN_CHANNEL_NUM ?DUP 
IF 
BC_FREE (Release the B-channel) 
ENDIF 
#NORMAL_CLEARING SETUP_CAUSE 
T." ----> Send RELEASE message " TCR 
M#REL MESSAGE> I#CAUSE <SEND (Send RELEASE message) 
19 NEW_STATE 
}ACTION 

M#REL ?L3_VALID_MSG (Network releases call) 
ACTION{ 
?CN_CHANNEL_NUM BC_FREE (Release B channel & Call Reference) 
*DEC ->C_CAUSE_VALUE @ SETUP_CAUSE 
T." ----> Send RELEASE COMPLETE message " TCR 
M#REL_COM MESSAGE> I#CAUSE <SEND 
CLEAR_CR (Clear call reference) 
0 NEW_STATE 
}ACTION 
)STATE 

4 STATE_INIT { 
2 CLEAR_KEY 
)STATE_INIT
4  STATE{  ( Call delivered )
   M#CONN ?L3_VALID_MSG  ( Call is connected )
   ACTION{
      ?CN_CHANNEL_NUM BC_CONN  ( Channel is connected )
      COUNTER3 @  ( Option to send CONNECT ACK )
      IF
         T." ----> Send CONNECT ACKNOWLEDGE message " TCR
         M#CONN_ACK MESSAGE> <SEND  ( Send a CONNECT ACK message )
      ENDIF
      10 NEW_STATE
   }ACTION

   M#DISC ?L3_VALID_MSG  ( Call is rejected )
   ACTION{
      ?CN_CHANNEL_NUM BC_FREE  ( Release the B-channel )
      #NORMAL_CLEARING SETUP_CAUSE
      T." ----> Send RELEASE message " TCR
      M#REL MESSAGE> I#CAUSE <SEND  ( Send RELEASE message )
      19 NEW_STATE
   }ACTION

   M#REL ?L3_VALID_MSG  ( Network releases call )
   ACTION{
      ?CN_CHANNEL_NUM BC_FREE  ( Release B channel & Call Reference )
      *DEC -> C_CAUSE_VALUE @ SETUP_CAUSE
      T." ----> Send RELEASE COMPLETE message " TCR
      M#REL_COM MESSAGE> I#CAUSE <SEND
      CLEAR_CR ( Clear call reference & connection handler )
      0 NEW_STATE
   }ACTION

}STATE

8  STATE_INIT{
   CLEAR_KEYS
   " DISCONNECT" 2 LABEL_KEY
}STATE_INIT

8  STATE{  ( Connect Request )
   UF2 ?KEY  ( Initiate call release )
   ACTION{
      #NORMAL_CLEARING SETUP_CAUSE
      T." ----> Send DISCONNECT message " TCR
      M#DISC MESSAGE> I#CAUSE <SEND  ( Send DISCONNECT message )
      RING_OFF  ( Turn the ring off )
      11 NEW_STATE
   }ACTION

   M#CONN_ACK ?L3_VALID_MSG  ( Received a CONNECT ACK message )
   ACTION{
      ?CN_CHANNEL_NUM BC_CONN  ( Connect the B channel )
      RING_OFF  ( Turn the ring off )
      10 NEW_STATE
   }ACTION
M#DISC ?L3_VALID_MSG  ( Network initiates call release )
ACTION{  
  ?CN_CHANNEL_NUM BC_FREE  ( Release B channel )  
  #NORMAL_CLEARING SETUP_CAUSE  
  T." ----> Send RELEASE message " TCR  
  M#REL MESSAGE> I#CAUSE <SEND  ( Send RELEASE message )  
  RING_OFF  ( Turn the ring off )  
  19 NEW_STATE  
}ACTION

M#REL ?L3_VALID_MSG  ( Network releases call )
ACTION{  
  ?CN_CHANNEL_NUM BC_FREE  ( Release B channel )  
  *DEC ->C_CAUSE_VALUE @ SETUP_CAUSE  
  T." ----> Send RELEASE message " TCR  
  M#REL_COM MESSAGE> I#CAUSE <SEND  ( Send RELEASE message )  
  RING_OFF  ( Turn the ring off )  
  0 NEW_STATE  
}ACTION

#USER1 ?CN_TIMER  ( Ringing timer is set )
ACTION{  
  RING_ON  
}ACTION
}

10 STATE_INIT{  
  " DISCONNECT" 2 LABEL_KEY  
}STATE_INIT

10 STATE{  
  ( Active )  
  UF2 ?KEY  ( User initiates call release )
ACTION{  
  #NORMAL_CLEARING SETUP_CAUSE  
  T." ----> Send DISCONNECT message " TCR  
  M#DISC MESSAGE> I#CAUSE <SEND  ( Send a DISCONNECT message )  
  11 NEW_STATE  
}ACTION

M#DISC ?L3_VALID_MSG  ( Remote party releases the call )
ACTION{  
  ?CN_CHANNEL_NUM BC_FREE  ( Release B channel )  
  #NORMAL_CLEARING SETUP_CAUSE  
  T." ----> Send RELEASE message " TCR  
  M#REL MESSAGE> I#CAUSE <SEND  ( Send RELEASE message )  
  19 NEW_STATE  
}ACTION
M#REL ?L3 VALID MSG  ( Remote party releases the call )
ACTION[
  ?CN_CHANNEL_NUM BC_FREE  ( Release B channel )
  *DEC ->C_CAUSE_VALUE @ SETUP_CAUSE
  T." ----> Send RELEASE COMPLETE message " TCR
  M#REL COM MESSAGE> I#CAUSE <SEND  ( Send RELEASE COMPLETE message )
  CLEAR CR  ( Clear call reference & connection handler )
  0 NEW_STATE
]ACTION
]STATE

11 STATE_INIT{
  2 CLEAR_KEY
]STATE

11 STATE{  ( Disconnect Request )
M#REL ?L3 VALID MSG  ( Network releases call )
ACTION[
  ?CN_CHANNEL_NUM ?DUP  ( Release allocated channel )
  IF
    BC_FREE
  ENDIF
  *DEC ->C_CAUSE_VALUE @ SETUP_CAUSE
  T." ----> Send RELEASE COMPLETE message " TCR
  M#REL_COM MESSAGE> I#CAUSE <SEND  ( Send RELEASE COMPLETE message )
  CLEAR_CR  ( Clear call reference & connection handler )
  0 NEW_STATE
]ACTION
]STATE

19 STATE_INIT{
  2 CLEAR_KEY
]STATE

19 STATE{  ( Release Request )
M#REL COM ?L3 VALID MSG  ( Release complete from network )
ACTION[
  CLEAR CR  ( Clear call reference & connection handler )
  CN DEALLOC  ( deallocate connection )
  0 NEW_STATE
]ACTION
]STATE
21.6 X.25 PLP Monitor/Emulation

The TEST1_X25.F and TEST2_X25.F test scripts run on the ISDN Basic Rate D-Channel with layer 3 X.25 PLP Emulation. The X25P_STAT.F test script runs on the ISDN Basic Rate D-Channel Monitor.

TEST1_X25.F

This test script generates call/clears for 100 seconds. Enter function key f1 to start the test. The COUNTER variable contains the number of calls done.

```
 pounded

 TCLR ( Initialize test script dictionary )

 ( STATE_INIT 0 is executed when the Test Script execution is started. )
 ( It is used to label function key f1 and move to the TestKeys topic. )
 ( Configure Layer 2 LAP D link 0 to use SAPI 16 for X.25 packet traffic. )
 ( Configure X.25 logical channels CH1 to CH8. )

 0 STATE_INIT{ "START TEST" 1 LABEL_KEY ( Label f1 key )
    2 CLEAR_KEY 3 CLEAR_KEY 4 CLEAR_KEY
    5 CLEAR_KEY 6 CLEAR_KEY 7 CLEAR_KEY 8 CLEAR_KEY
    DROP_TEST ( Do not show Test Script Window )
    "TestKeys" SET_CURR_TOPIC ( Move to TestKeys topic )
    0 SA ( Use Link 0 )
    16 SAPI ( SAPI 16 used )
   CH1 SVC 1 =LCN 0 =LCNCES ( CH1 LCN parameters )
   CH2 SVC 2 =LCN 0 =LCNCES ( CH2 LCN parameters )
   CH3 SVC 3 =LCN 0 =LCNCES ( CH3 LCN parameters )
   CH4 SVC 4 =LCN 0 =LCNCES ( CH4 LCN parameters )
   CH5 SVC 5 =LCN 0 =LCNCES ( CH5 LCN parameters )
   CH6 SVC 6 =LCN 0 =LCNCES ( CH6 LCN parameters )
   CH7 SVC 7 =LCN 0 =LCNCES ( CH7 LCN parameters )
   CH8 SVC 8 =LCN 0 =LCNCES ( CH8 LCN parameters )
 }STATE_INIT

 ( STATE 0 is used to wait for the f1 function key to be entered. )
 ( Then the link connection will be established and a SABME transmitted. )
```
0 STATE( UF1 ?KEY
  ACTION( T. " TEST STARTING" TCR
    " Test Starting. Sending Calls/Clears for 100 Seconds" W.NOTICE
    1 CLEAR_KEY ( Clear f1 key )
    DL_ESTABLISH ( Establish link connection )
    1 NEW_STATE )ACTION
  )STATE

( STATE 1 will wait for the UA response indicating that the link connection )
( is established and that layer 3 traffic can commence. )
( Then a Restart Packet will be transmitted. )

1 STATE( R#UA ?RX_FRAME
  ACTION( RESTART ( Send Restart packet )
    2 NEW_STATE )ACTION
  )STATE

( STATE 2 will wait for the Restart confirm packet, then will generate call )
( request packets on Channels 1 to 8. The COUNTER variable is used to count )
( the number of calls done will be initialized to a value of 8. )
( Timer 101 is started with a duration of 100 seconds. )
( The test will go to state 3 for the next events. )

2 STATE( R*RESTARTCONF 1 ?RX
  ACTION( CH1 CALL CH2 CALL CH3 CALL CH4 CALL
    CH5 CALL CH6 CALL CH7 CALL CH8 CALL
    8 COUNTER ! ( Start count of calls established )
    101 1000 START_TIMER ( Start 100 sec timer )
    3 NEW_STATE )ACTION
  )STATE

( STATE 3 continues to generate calls and clears until timer 101 expires. )
( Then a Restart request is generated and the test goes to state 4. )

3 STATE( R*CALLCONF 1 ?RX
  ACTION( CLEAR )ACTION
    R*CLEARCONF 1 ?RX ( After each LCN cleared, make new call )
    ACTION( CALL
      1 COUNTER +! )ACTION

    101 ?TIMER ( Keep making calls for 100 seconds )
    ACTION( RESTART ( Send Restart after 100 Seconds )
      4 NEW_STATE )ACTION
  )STATE

( State 4 will wait for the Restart confirm packet, then disconnect the )
( layer 2 connection. The test will then go to state 5. )

4 STATE( R*RESTARTCONF 1 ?RX
  ACTION( DL_RELEASE ( Disconnect Layer 2 )
    5 NEW_STATE )ACTION
  )STATE
( State 5 will wait for the layer 2 UA response, then generate results )
( for the test. The total number of calls and calls per second will be )
( displayed in the Test Script Window. The test manager then will be )
( turned off and the TestScript Topic will become the current topic. )

5 STATE[ R#UA ?RX_FRAME ( Wait for UA, Layer 2 disconnected )
   ACTION[ T.¨ Test Finished. ¨ TCR
   ¨ W.NOTICE ( Clear Notice line on screen )
   SHOW_TEST CR ( Show Script window on screen )
   OPEN_TEST ( Display output in Script window )
   WHI_FG SET_ATTR ( Display using White characters )
   COUNTER @ .¨ Calls in 100 seconds. ¨
   COUNTER @ 100 U/
     WHERE 1 - THERE .¨ .¨
     .¨ Calls per second.¨ CR
   " UFl" 1 LABEL_KEY
   " TestScript" SET_CURR_TOPIC ( Move to TestScript Topic )
   TM_STOP ( Stop Test manager )
   )ACTION
)

STATE
**TEST2_X25.F**

Enter function key f1 to start the test; function key f2 to stop the test.

```
TCLR ( Initialize test script dictionary )

( Sequence 0 will generate data packets on the current logical channel )
( until the window is closed. )

0 SEQ{ BEGIN
    DATA                  ( send data packets )
    WINDOW? 0= UNTIL ( until window is closed )
}SEQ

STATE_INIT{ "Start Test" 1 LABEL_KEY ( Label f1 key )
            2 CLEAR_KEY  3 CLEAR_KEY  4 CLEAR_KEY
            5 CLEAR_KEY  6 CLEAR_KEY  7 CLEAR_KEY  8 CLEAR_KEY
            DROP_TEST ( Do not show Test Script Window )
            "TestKeys"  SET_CURR_TOPIC ( Move to TestKeys topic )
            0 SA ( Use Link 0 )
            16 SAPI ( SAPI 16 used )
        CH1 SVC  1 =LCN  0 =LCNCES ( CH1 LCN parameters )
        CH2 SVC  2 =LCN  0 =LCNCES ( CH2 LCN parameters )
        CH3 SVC  3 =LCN  0 =LCNCES ( CH3 LCN parameters )
        CH4 SVC  4 =LCN  0 =LCNCES ( CH4 LCN parameters )
        CH5 SVC  5 =LCN  0 =LCNCES ( CH5 LCN parameters )
        CH6 SVC  6 =LCN  0 =LCNCES ( CH6 LCN parameters )
        CH7 SVC  7 =LCN  0 =LCNCES ( CH7 LCN parameters )
        CH8 SVC  8 =LCN  0 =LCNCES ( CH8 LCN parameters )
}STATE_INIT
```
STATE 0 is used to wait for the f1 function key to be entered. Then the link connection will be established and a SABME transmitted.

STATE 0
ACTION T. "TEST STARTING" TCR
   "Test Starting. Sending Calls on LCN 1 to 8" w.NOTICE
  1 CLEAR_KEY (Clear f1 key)
  DL_ESTABLISH (Establish link connection)
  1 NEW_STATE }ACTION

STATE 1 will wait for the UA response indicating that the link connection is established and that layer 3 traffic can commence. Then a Restart Packet will be transmitted.

STATE 1
ACTION R#UA ?RX_FRAME (Wait until UA received)
   2 NEW_STATE }ACTION

STATE 2 will wait for the Restart confirm packet, then will generate call request packets on Channels 1 to 8. The COUNTER variable is used to count the number of cycles through making calls, sending data, and clearing. The test will go to state 3 for the next events.

STATE 2
ACTION R*RESTARTCONF 1 ?RX (Start generating calls)
   0 COUNTER ! (Clear count of calls)
  3 NEW_STATE }ACTION

STATE 3 will wait for all 8 Call connected packets, then generate data packets on all 8 channels.

STATE 3
ACTION R*CALLCON 1 ?RX (Wait for call connected packet.)
   1 COUNTER ++ COUNTER @ 8 =
IF  ( Check for 8 call connected packets )
   ( Then generate data packets on all 8 channels )
   " Sending Data on LCN 1 to 8" W.NOTICE
   CH1 0 RUN_SEQ        ( send data packets on CH1 )
   CH2 0 RUN_SEQ        ( send data packets on CH2 )
   CH3 0 RUN_SEQ        ( send data packets on CH3 )
   CH4 0 RUN_SEQ        ( send data packets on CH4 )
   CH5 0 RUN_SEQ        ( send data packets on CH5 )
   CH6 0 RUN_SEQ        ( send data packets on CH6 )
   CH7 0 RUN_SEQ        ( send data packets on CH7 )
   CH8 0 RUN_SEQ        ( send data packets on CH8 )
   " Stop Test" 2 LABEL_KEY
4 NEW_STATE
ENDIF
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
6 STATE{ R#UAPRX_FRAME
  
  ACTION{ T." Test Finished" TCR TM_STOP
          " TestScript" SET_CURR_TOPIC ( Move to TestScript Topic )
  }ACTION

}STATE
**X25P_STAT.F**

(--------------------------------------------------------)

(X25P_STAT.F : X.25 ANALYZER Test Manager Scenario)

( Collect and display X.25 Packet statistics)

( COUNTER1 to COUNTER30 collect counts.)

(f1 = Display Statistics.)

(f2 = Display Data.)

(f3 = Clear Statistic counts.)

(--------------------------------------------------------)

TCLR

( CLEAR Test manager dictionary )

WAKEUP_ON

( Wakeup the test manager to display statistics on startup )

0 STATE_INIT(

" Show Stat" 1 LABEL_KEY

" Show Data" 2 LABEL_KEY

" Clear" 3 LABEL_KEY

" Stop Test" 4 LABEL_KEY )STATE_INIT

0 STATE(

( Define state 0 to collect statistics and ck events )

R*CALLREQ 1 ?RX ACTION(

OPEN_USER

1 COUNTER10 +!

2 48 THERE COUNTER10 @ W.

CLOSE_WINDOW )ACTION

( Call request received )

( Open user display window )

( Update Call request counter )

( Display Call request count )

( Close display window )

R*CALLCON 1 ?RX ACTION(

OPEN_USER

1 COUNTER11 +!

3 48 THERE COUNTER11 @ W.

CLOSE_WINDOW )ACTION

( Call connect received )

( Open user display window )

( Update Call connect counter )

( Display Call connect count )

( Close display window )

R*CLEARREQ 1 ?RX ACTION(

OPEN_USER

1 COUNTER12 +!

4 48 THERE COUNTER12 @ W.

CLOSE_WINDOW )ACTION

( Clear request received )

( Open user display window )

( Update Clear request counter )

( Display Clear request count )

( Close display window )

R*CLEARCONF 1 ?RX ACTION(

OPEN_USER

1 COUNTER13 +!

5 48 THERE COUNTER13 @ W.

CLOSE_WINDOW )ACTION

( Clear confirm received )

( Open user display window )

( Update Clear confirm counter )

( Display Clear confirm count )

( Close display window )
R*RESTARTREQ  1 ?RX ACTION{  
OPEN_USER  
1 COUNTER14 +!  
6 48 THERE COUNTER14 @ W.  
CLOSE_WINDOW  )ACTION  
( Restart request received  )  
( Open user display window  )  
( Update Restart request count  )  
( Display Restart request count  )  
( Close display window  )  

R*RESTARTCONF  1 ?RX ACTION{  
OPEN_USER  
1 COUNTER15 +!  
7 48 THERE COUNTER15 @ W.  
CLOSE_WINDOW  )ACTION  
( Restart confirm received  )  
( Open user display window  )  
( Display Restart confirm count  )  
( Close display window  )  

R*RESETREQ  1 ?RX ACTION{  
OPEN_USER  
1 COUNTER16 +!  
8 48 THERE COUNTER16 @ W.  
CLOSE_WINDOW  )ACTION  
( Reset request received  )  
( Open user display window  )  
( Update Reset request count  )  
( Display Reset request count  )  
( Close display window  )  

R*RESETCONF  1 ?RX ACTION{  
OPEN_USER  
1 COUNTER17 +!  
9 48 THERE COUNTER17 @ W.  
CLOSE_WINDOW  )ACTION  
( Reset confirm received  )  
( Open user display window  )  
( Display Reset confirm count  )  
( Close display window  )  

R*DATAP  1 ?RX ACTION{  
OPEN_USER  
1 COUNTER18 +!  
10 48 THERE COUNTER18 @ W.  
CLOSE_WINDOW  )ACTION  
( Data packet received  )  
( Open user display window  )  
( Display Data packet count  )  
( Close display window  )  

R*RRP  1 ?RX ACTION{  
OPEN_USER  
1 COUNTER19 +!  
11 48 THERE COUNTER19 @ W.  
CLOSE_WINDOW  )ACTION  
( RR packet received  )  
( Open user display window  )  
( Display RR packet count  )  
( Close display window  )
UF1 ?KEY  WAKEUP OR  ( f1 key or TM_RUN wakeup events )
CF1 ?KEY  OR
ACTION(  ( Open and show user display window )
    POP_USER  ( Open user display window )
    CLEAR_TEXT  WHI_FG PAINT  ( Clear screen text and color )
    " X.25 Packet Statistics"  WHI_FG  15 70 0 5 CYA_FG  HEADLINE
13 10 THERE  W." f1 = Show Statistics ,
W." f2 = Show Data ,
W." f3 = Clear Statistics "
2 30 THERE  W." CALL REQUEST  = "  COUNTER10  @ W.
3 30 THERE  W." CALL CONNECT  = "  COUNTER11  @ W.
4 30 THERE  W." CLEAR REQUEST  = "  COUNTER12  @ W.
5 30 THERE  W." CLEAR CONFIRM  = "  COUNTER13  @ W.
6 30 THERE  W." RESTART REQUEST  = "  COUNTER14  @ W.
7 30 THERE  W." RESTART CONFIRM  = "  COUNTER15  @ W.
8 30 THERE  W." RESET REQUEST  = "  COUNTER16  @ W.
9 30 THERE  W." RESET CONFIRM  = "  COUNTER17  @ W.
10 30 THERE  W." DATA PACKET  = "  COUNTER18  @ W.
11 30 THERE  W." RR PACKET  = "  COUNTER19  @ W.
CLOSE_WINDOW
)ACTION

UF2 ?KEY  CF2 ?KEY OR
ACTION(  SHOW_DATA )ACTION  ( Show data window )

UF3 ?KEY  CF3 ?KEY OR  ACTION(  ( Clear statistic counters )
    0 COUNTER1  ! 0 COUNTER2  ! 0 COUNTER3  ! 0 COUNTER4  ! 0 COUNTER5  !
    0 COUNTER6  ! 0 COUNTER7  ! 0 COUNTER8  ! 0 COUNTER9  ! 0 COUNTER10  !
    0 COUNTER11  ! 0 COUNTER12  ! 0 COUNTER13  ! 0 COUNTER14  ! 0 COUNTER15  !
    0 COUNTER16  ! 0 COUNTER17  ! 0 COUNTER18  ! 0 COUNTER19  ! 0 COUNTER30  !
)ACTION

UF4 ?KEY  CF4 ?KEY OR  ACTION(  TM_STOP  CLEAR_TEXT  WHI_FG PAINT
    " UF1"  1 LABEL_KEY
    " UF2"  2 LABEL_KEY
    " UF3"  3 LABEL_KEY
    " UF4"  4 LABEL_KEY
)ACTION
)STATE  ( End of state 0 )

A.1 The Origins of ISDN

Traditionally, voice communication was the only service supported by the telecommunication companies. The advent of data communications required the provisioning of dedicated data lines along with a plethora of interface types and speeds. Other services, such as facsimile and switched data have become more and more important to both the subscriber and the telephone company. In the future, new services, such as switched video and high speed data, will become telecommunication company offerings.

In order to accommodate the vast number of services that will be offered to the consumer, the telecommunications companies realized that their services must be integrated so as to provide a standard access to the telecommunications network, in much the same way that the power company provides a standard wall plug for power access. The ISDN (Integrated Services Digital Network) will be the structure to provide these different types of telecommunications services over a common physical user interface.

ISDN was laid down by the International Telegraph and Telephone Consultative Committee (CCITT) in order to provide the elements of a universal telecommunications network, namely the ISDN.

A.2 ISDN Access

There are two different types of ISDN access. The first is BRA (Basic Rate Access) consisting of two 64 kbit/s channels known as B–Channels, and a 16 kbit/s signalling channel known as the D–Channel (see Figure A–1). The B–Channels can carry voice, data, or image information.

The D–Channel carries the signalling information for the B–Channels and can carry low speed packetized information. This configuration is known as 2B+D. The bit rate over a Basic Rate access is 64 kbit/s + 64 kbit/s + 16 kbit/s = 144 kbit/s. The addition of framing bits and other support bits brings the line speed up to 192 kbit/s on the S/T interface or 160 kbit/s on the U interface (see Section A.3 for a description of the S/T and U interfaces).
The second type of access is PRA (Primary Rate Access). Primary Rate access in North America consists of 23 64 kbit/s B-Channels and a 64 kbit/s D-Channel. The configuration is known as 23B+D (see Figure A-2). The bit rate over the Primary Rate Access is $24 \times 64\,\text{kbit/s} = 1.536\,\text{Mbit/s}$. The addition of framing bits brings the line speed up to 1.544 Mbit/s. In Europe, Primary Rate access consists of 30 B-Channels and a D-Channel (and a framing channel of 64 kbit/s). This configuration is known as 30B+D. The line speed in this configuration is 2.048 Mbps.

**Figure A-1 Basic Rate Configuration**

<table>
<thead>
<tr>
<th>Voice</th>
<th>( \rightarrow )</th>
<th>B1</th>
<th>64 kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>+ framing bits = 192 kbit/s</td>
</tr>
<tr>
<td>Data</td>
<td>( \rightarrow )</td>
<td>B2</td>
<td>64 kbit/s</td>
</tr>
<tr>
<td>Signalling</td>
<td>( \rightarrow )</td>
<td>D</td>
<td>16 kbit/s</td>
</tr>
</tbody>
</table>

**Figure A-2 Primary Rate Configuration**

<table>
<thead>
<tr>
<th>Voice</th>
<th>( \rightarrow )</th>
<th>B1</th>
<th>64 kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>+ framing bits = 1.544 Mbps</td>
</tr>
<tr>
<td>Data</td>
<td>( \rightarrow )</td>
<td>B2</td>
<td>64 kbit/s</td>
</tr>
<tr>
<td></td>
<td>( \rightarrow )</td>
<td>B3</td>
<td>64 kbit/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>+ framing bits = 1.544 Mbps</td>
</tr>
<tr>
<td>Voice</td>
<td>( \rightarrow )</td>
<td>B23</td>
<td>64 kbit/s</td>
</tr>
<tr>
<td>Signalling</td>
<td>( \rightarrow )</td>
<td>D</td>
<td>16 kbit/s</td>
</tr>
</tbody>
</table>
A.3 The ISDN Network

An example of an ISDN network is shown in Figure A-3. At the user 1 side there are a number of pieces of equipment connected to a Basic Rate access line. This equipment is connected to a PABX, which is in turn connected to an ISDN switch via a Primary Rate facility.

A TE1 (terminal equipment 1) is an ISDN terminal and can have both data and voice capabilities. A TE2 is a non-ISDN terminal such as a telephone or a computer terminal. The TE2 is connected to the S reference point (the S/T Bus) through a TA (terminal adapter). The S/T Bus can support up to eight devices. The S/T Bus is connected to an NT2. The NT2 (network termination 2) provides interface termination, layer 2 and 3 protocol handling, multiplexing, switching and concentration. A typical example of an NT2 is a PABX.

The NT2 is connected to an NT1 (network termination 1) through the T interface. The NT1 terminates the signal from the network and provides layer 1 maintenance capabilities such as loopback and line buildout. The line from the subscriber is terminated at the telephone office by an LT (line termination).

At the switch, signalling information, controlling the routing of calls between switches, is sent via the common channel signalling number 7 system (CCS7) while data is sent via separate carriers. Signalling information is gathered or transmitted at switches and is transferred at signalling transfer points (STP). Signalling information can also be transferred directly between switches.
Figure A-3 ISDN Network
A.4 Channel Structure

The D-Channel is layered according to the OSI (open systems interconnection) model of data communications layering (see Figure A-4).

Layer 1 provides the physical, mechanical, and electrical conditions for transmission of the Basic Rate or Primary Rate signal.

Layer 2, or the data link layer, contained in the D-Channel, provides end-to-end connection and information transfer.

Layer 3 carries the signalling information for the B-Channels as well as user-user information over the D-Channel.

Each layer has a management entity which manages the flow of communication and the transfer of data at its layer, along with communication between adjacent layers (see Figure A-5).

![Figure A-4 OSI Model of the D-Channel](image)

![Figure A-5 Management Layers](image)
A.5 Communication Between Layers in the D-Channel

The OSI model for communication between layers requires that primitives form the basis of communication between adjacent layers inside equipment. Primitives strictly define the interface between layers and detail the actions that can occur between layers.

Consequently, the ISDN protocol provides communication primitives between the physical layer and the data link layer, and between the data link layer and the network layer.

There are four different types of primitives between layers:
- Request
- Indication
- Response
- Confirm

The primitives can contain additional information and parameters.

The primitives between layer 1 and layer 2 are:
- **PH-DATA**: data transfer (request, indication)
- **PH-ACTIVATE**: establish a layer 1 connection (request, indication)
- **PH-DEACTIVATE**: release a layer 1 connection (request, indication)

The primitives between layer 1 and its management layer are:
- **MPH-ACTIVATE**: layer 1 is activated (indication)
- **MPH-DEACTIVATE**: layer 1 is deactivated (indication)
- **MPH-INFORMATION**: provides information indicating whether there is sufficient power, and whether the terminal is connected (indication)
- **MPH-ERROR**: error in layer 1 (indication, response)

For example, to initiate a layer 1 connection, the layer 2 sends a PH–ACTIVATE request to layer 1. If the activation is successful, the layer 1 sends a PH–ACTIVATE indication to layer 2.

The primitives between layer 2 and 3 are:
- **DL-ESTABLISH**: establish a link (request, indication, confirmation)
- **DL-RELEASE**: release a link (request, indication, confirmation)
- **DL-UNIT-DATA**: data transfer (request, indication)
- **DL-DATA**: data transfer, acknowledged operation (request, indication)

For example, in order for layer 3 to establish a connection, send a message, and release the connection, it must send a DL–ESTABLISH request to layer 2. After servicing the request, layer 2 in turn responds with a DL–ESTABLISH confirmation. Layer 3 sends data using either a DL–UNIT-DATA request, for unacknowledged information transfer, or DL–DATA, for acknowledged information transfer. At the end of the session, layer 3 indicates to layer 2 to release the connection by issuing a DL–RELEASE request. Layer 2 responds with a DL–RELEASE confirmation.
A number of primitives are also used between layer 2 and its management entity. These are:

- **MDL-ASSIGN**: assign a TEI terminal endpoint identifier to the link (request, indication)
- **MDL-REMOVE**: remove the association between a link and a TEI (request)
- **MDL-UNIT-DATA**: send management control information (request, indication)
- **MDL-ERROR**: error diagnostics (indication, response)

If a TEI has not been assigned, layer 2 passes an MDL-ASSIGN indication to the layer 2 management entity upon receipt of a DL-ESTABLISH request. If a TEI is required, the management entity gives an MDL-ASSIGN request to layer 2. Layer 2 will proceed to request a TEI from the network.

### A.6 Layer 2 — Data Link Layer (LAPD)

The data link layer implements the link access protocol on the D-Channel (LAPD) and allows the interchange of frames between terminals and the NT2, between the NT2 and the exchange, or between the terminal and the exchange.

The data link layer provides the following services:
- Multiplexing of several channels on the same D-Channel
- Detection of transmission, formatting and operating errors on the data link
- Retransmission of faulty frames
- Flow control

### Addressing

A layer 3 communication entity is identified by two elements which comprise the CEI (connection endpoint identifier). The first of these is the SAPI (service access point identifier). The second is the CES (connection endpoint suffix).

The SAPI identifies a point through which data link services are provided to the layer 3 entity. There are three commonly used types of data link services. The first of these are the management procedures. These procedures have a SAPI of 63. These procedures are used for TEI assignment and control, and other management procedures.

The second type of data link services are the call control procedures. These procedures are identified by a SAPI of 0. These procedures control calls placed over the B-Channels.

The third type of service is the packet communication service which is identified by a SAPI of 16. This service sends data over the D-Channel. For example, the excess capacity in a D-Channel can be used for data communication between low speed terminals (up to 9600 bps).

The CES identifies the layer 3 entity connected to a particular service. For example, in Figure A-6, terminal A is connected to the packet communication service. Since there is more than one terminal connected to this service, terminal A is identified by a number – the CES. The CEI is thus the SAPI (16 in this case) plus the CES (2 in this case).
The layer 3 entity uses the CEI to identify or address its connection to layer 2. The layer 2 in turn identifies this connection by the DLCI (data link connection identifier). The DLCI is contained within the layer 2 frame. The DLCI is composed of two parts: the SAPI and the TEI (terminal endpoint identifier).

The TEI uniquely identifies the terminal within a service access point. For example, in Figure A–6, terminal A is identified by a TEI of 64.

![Diagram of TEI/SAPI Relationship](image)

**Figure A–6 TEI/SAPI Relationship**

### Establishing a Layer 2 Link

Figure A–7 shows the data flow in a layer link establishment. The S/T Bus has already been activated. The first action the user equipment performs is to request a TEI. The network responds with a TEI assignment to be used in future messages.

The user then sends a SABME (set asynchronous balanced mode extended). This command prepares both sides for information transfer using numbered frames that cycle through the values 0 through 127. The network responds with a UA (unnumbered acknowledge) frame.

Information frames are then exchanged and are acknowledged using RR (receiver ready) frames. At the end of information transfer, the user issues a DISC (disconnect) message to the network. The network responds with a UA frame.
Figure A-7 Data Flow in a Link Establishment
LAPD Frames

There are three different types of LAPD frames:

- Information
- Supervisory
- Unnumbered information

Information frames carry information between the ends of the communication. They contain sequence numbers and thus require an acknowledgement from the other end.

Unnumbered frames carry additional data link control functions. They do not carry sequence numbers and thus do not require acknowledgement of transmission. Unnumbered frames are used for several functions. First, they set the modulus mode. Second, they carry frame reject, frame acknowledge, disconnect, and other data link control functions. Third, they are used by layer 2 to carry DL–UNIT–DATA information from the layer 3. Unnumbered information frames are also used by the layer 2 management entity to carry TEI request and assign frames as a result of MDL–ASSIGN procedures from the management entity.

The supervisory frames perform supervisory control functions such as information frame acknowledgement, retransmission requests, and temporary suspension of transmission.

Information and supervisory frames are sequenced using either a modulo 8 (normal) or modulo 128 (extended) scheme. The sequencing ensures that frames are sent in the proper order. If retransmission of a frame is required, specifying the sequence number identifies the frame to be retransmitted. Note that modulo 8 operation is not supported by the 1988 Blue Book.
See Table A–1 for a list of the LAPD frames for modulo 8 operation, and Table A–2 for a list of the frames for modulo 128 operation.

### Table A–1 Frames for Modulo 8 Operation

<table>
<thead>
<tr>
<th>Format</th>
<th>Commands</th>
<th>Responses</th>
<th>Encoding</th>
<th>Octet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Information Transfer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>I (information)</td>
<td>N(S)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N(R)</td>
<td>P</td>
<td>5</td>
</tr>
<tr>
<td>Supervisory</td>
<td>RR (receive ready)</td>
<td>N(R)</td>
<td>0 0 0 0 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P/F</td>
<td>0 1 0 1 1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>RNR (receive not ready)</td>
<td>N(R)</td>
<td>0 0 0 0 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P/F</td>
<td>0 1 0 1 1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>REJ (reject)</td>
<td>N(R)</td>
<td>1 0 0 0 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P/F</td>
<td>1 1 1 1 1</td>
<td>5</td>
</tr>
<tr>
<td>Unnumbered</td>
<td>SABM (set asynchronous balance mode)</td>
<td>0 0 1</td>
<td>P 1 1 1 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DM (disconnect</td>
<td>0 0 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mode)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UI (unnumbered information)</td>
<td>0 0 0</td>
<td>P 0 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disc (disconnect)</td>
<td>0 1 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UA (unnumbered</td>
<td>0 1 1 0 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>acknowledge)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRMR (frame</td>
<td>0 1 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reject)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>XID (exchange</td>
<td>1 0 1 0 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>identification)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table A–2 Frames for Modulo 128 Operation

<table>
<thead>
<tr>
<th>Format</th>
<th>Commands</th>
<th>Responses</th>
<th>Encoding</th>
<th>Octet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Information Transfer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>I (information)</td>
<td>N(S)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N(R)</td>
<td>P</td>
<td>5</td>
</tr>
<tr>
<td>Supervisory</td>
<td>RR (receive ready)</td>
<td>N(R)</td>
<td>0 0 0 0 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P/F</td>
<td>0 1 0 1 1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>RNR (receive not ready)</td>
<td>N(R)</td>
<td>0 0 0 0 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P/F</td>
<td>0 1 0 1 1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>REJ (reject)</td>
<td>N(R)</td>
<td>0 0 0 0 0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P/F</td>
<td>1 0 1 1 1</td>
<td>5</td>
</tr>
<tr>
<td>Unnumbered</td>
<td>SABME (set asynchronous balance mode extended)</td>
<td>0 1 1</td>
<td>P 1 1 1 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DM (disconnect</td>
<td>0 0 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mode)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UI (unnumbered information)</td>
<td>0 0 0</td>
<td>P 0 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disc (disconnect)</td>
<td>0 1 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UA (unnumbered</td>
<td>0 1 1 0 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>acknowledge)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRMR (frame</td>
<td>1 0 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reject)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>XID (exchange</td>
<td>1 0 1 0 1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>identification)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The structure of the LAPD frame is shown in Figure A-8. The frame starts with a hex 7E flag.

![LAPD Frame Structure Diagram]

The next two bytes are the address. The first byte contains the SAPI and the C/R bit (command/response bit) which indicates whether the frame is a command or a response. The TEI is contained in the second byte.

The next one or two bytes is the control field which identify the type of the frame. When applicable, the control field contains sequence numbers. The control field structure is also shown in Tables A-1 and A-2.

Next, is the information field which can be of variable length up to a maximum of 260 bytes. The information field is not present if the frame carries only data link control commands.

Finally, is a frame check sequence and a closing flag.

A.7 Layer 3 – Network Layer

The network layer provides the services to establish, maintain, and terminate calls across an ISDN between communicating entities.
Operation of the Network Layer

Figure A-9 shows a simple ISDN call for voice communication. When the call is commenced by the user picking up a phone and dialing a number, the terminal sends a SETUP message to the network over an already established data link. The SETUP message contains a unique value identifying the call, known as the call reference value, and is used in all communication regarding this call.

In addition to the call reference value, the SETUP message also contains information such as the called number, the type of service to be used (voice, data, etc.), the channel to be used (B1, B2, or any channel), and the capabilities of the terminal.

When the network receives the called number, it sends a CALL PROCEEDING message back to the user and tries to connect to the called terminal. The network sends a SETUP message to the called terminal. If the called terminal is able to accept the call, it returns an ALERT message to the network. This message indicates to the network that it is accepting the call and is ringing. The network sends an ALERT message to the calling terminal to indicate that the far end is ringing.
When the called party picks up the phone, the terminal sends a CONNECT message to the network. The network in turn sends a CONNECT ACKNOWLEDGE message back to the called party. The network also sends a CONNECT message to the calling party to indicate a connection has been made. The calling party's terminal sends a CONNECT ACKNOWLEDGE back to the network.

At this point, the two parties can talk to each other.

When the called party hangs up, the terminal sends a DISCONNECT message to the network. The network sends a DISCONNECT message to the calling party and sends a RELEASE message to the called party. The called party's terminal sends a RELEASE COMPLETE message to confirm disconnection of the call. The calling party's terminal sends a RELEASE Message to the network. The network responds by sending a RELEASE COMPLETE message.

Structure of the Network Layer

The network layer information is carried in layer 2 information frames as shown in Figure A-10. The network layer packet is formatted as shown in Figure A-11.

![Figure A-10 Layer 2 Information Frames](image)

![Figure A-11 Network Layer Packet](image)

The first byte, the protocol discriminator, distinguishes between messages which are coded according to CCITT standards from other national message sets.

The second byte contains the length of the call reference value in bytes. The call reference value is contained in the third and successive bytes, and uniquely identifies the call at the User/Network interface. The first bit is the call reference flag. The call reference flag identifies the layer 3 message as coming from the side originating the call, or the destination side.
The next byte, after the call reference value, is the message type identifier. The message type is a general type command such as Setup, Alert or Call Proceeding.

Successive bytes are allocated to information elements, of which there can be zero, one, or more in a message. The information elements carry information relevant to the message. For example, a SETUP message might contain the terminal capabilities in one information element and the called number in another information element.

Information elements can be encoded in two different formats. Figure A-12 shows the single octet information element format. This format is used when one octet can hold the required information. Figure A-13 shows the variable length information element. In this case, the first octet is the information element identifier. The second octet is the length of the information element in octets. The third and successive octets carry the information of the information elements. To code this part of the information element, consult the CCITT Blue Book.

<table>
<thead>
<tr>
<th>1</th>
<th>identifier</th>
<th>contents</th>
</tr>
</thead>
</table>

**Figure A-12 Single Octet Information Element**

<table>
<thead>
<tr>
<th>0</th>
<th>Information Element Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Length of Information element (Octets)</td>
</tr>
<tr>
<td></td>
<td>Contents</td>
</tr>
</tbody>
</table>

**Figure A-13 Variable Length Information Element**
This appendix contains a brief introduction to X.25. For further information, the reader is advised to consult the CCITT X.25 (1984) Recommendation, Red Book VIII.3, Malaga–Torremolinos.

B.1 Purpose of the Recommendation

This recommendation was established to produce standards to facilitate international networking between various countries which have established public data networks providing packet switched data transmission services.

B.2 Types of X.25 Services

PVC (permanent virtual circuit) is the establishment of a permanent relationship between two users.

SVC (switched virtual circuit) is a relationship that is established only for the duration of a call. SVC requires a call setup phase, the delivery of data in sequences, and a call clearing phase to terminate the call.

B.3 Layer 3 – Network Level

The network layer performs basic multiplexing of data and allows many different virtual circuits to be operated over a single link layer. It is at this level that call setup and clearing occurs, as well as access to network provided facilities. Data transfer is accomplished with a mechanism that ensures that information is transferred in the correct sequence.

The network layer information is contained in packets which are carried in information frames. The packet information commences in the first byte following the control field of an I frame. This byte is commonly called octet one when referring to packet formats.

The packet structure discussed here is for layer 3 modulo 8, only as described in the 1984 Recommendation. The user should consult the Recommendation for modulo 128 structures.
Each packet contains a packet control header consisting of three octets. Table B-1 shows the format for the packet control header with the size of each field in bits.

<table>
<thead>
<tr>
<th>Octets</th>
<th>Bits</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8 7 6 5 4 3</td>
<td>General format identifier Logical channel group number</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Logical channel number</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Packet type identifier</td>
</tr>
</tbody>
</table>

Table B-1 Packet Header

**GFI (General Format Identifier)**

This field identifies the general format identifier of the data packet sequencing scheme, i.e. modulo 8 or modulo 128. Packet sequence numbers cycle between 0 and 7 in modulo 8, and 0 and 127 for modulo 128. The modulo is determined by examining bits 5 and 6 of the first octet of the packet.

Data packets contain a Q bit within the GFI. This is the eighth bit of octet 1 in the data packet header. When the Q (qualifier) bit is set to 1, the data packet being sent is not user data.

Data packets and call setup packets contain a D bit. This is the seventh bit of octet 1 in the packet header. When the D (delivery confirmation) bit is set, an acknowledgement from a distant subscriber is required.

**Logical Channel Group Number and Logical Channel Number**

These fields show information regarding the destination of a packet over a logical virtual channel in the network.
Packet Identifier

Each packet is identified in octet 3 according to Table B-2. A bit which is marked as X in this table can be set to a 0 or 1.

<table>
<thead>
<tr>
<th>Packet Type</th>
<th>Octet 3 Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8 7 6 5 4 3 2 1</td>
</tr>
<tr>
<td><strong>Call setup and clearing</strong></td>
<td></td>
</tr>
<tr>
<td>Incoming call</td>
<td>0 0 0 0 1 0 1 1</td>
</tr>
<tr>
<td>Call connected</td>
<td>0 0 0 0 1 1 1 1</td>
</tr>
<tr>
<td>Clear indication</td>
<td>0 0 0 1 0 0 1 1</td>
</tr>
<tr>
<td>DCE clear confirmation</td>
<td>0 0 0 1 0 1 1 1</td>
</tr>
<tr>
<td><strong>Data and interrupt</strong></td>
<td></td>
</tr>
<tr>
<td>DCE data</td>
<td>X X X X X X X 0</td>
</tr>
<tr>
<td>DCE interrupt</td>
<td>0 0 1 0 0 0 1 1</td>
</tr>
<tr>
<td>DCE interrupt confirmation</td>
<td>0 0 1 0 0 1 1 1</td>
</tr>
<tr>
<td><strong>Flow control and reset</strong></td>
<td></td>
</tr>
<tr>
<td>DCE RR (modulo 8)</td>
<td>X X X 0 0 0 0 1</td>
</tr>
<tr>
<td>DCE RR (modulo 128)</td>
<td>0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>DCE RNR (modulo 8)</td>
<td>X X X 0 0 1 0 1</td>
</tr>
<tr>
<td>DCE RNR (modulo 128)</td>
<td>0 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>DTE REJ (modulo 8)</td>
<td>X X X 0 1 0 0 1</td>
</tr>
<tr>
<td>DTE REJ (modulo 128)</td>
<td>0 0 0 0 1 0 0 1</td>
</tr>
<tr>
<td>Reset indication</td>
<td>0 0 0 1 1 0 1 1</td>
</tr>
<tr>
<td>DCE reset confirmation</td>
<td>0 0 0 1 1 1 1 1</td>
</tr>
<tr>
<td><strong>Restart</strong></td>
<td></td>
</tr>
<tr>
<td>Restart indication</td>
<td>1 1 1 1 1 0 1 1</td>
</tr>
<tr>
<td>DCE restart confirmation</td>
<td>1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td><strong>Diagnostic</strong></td>
<td></td>
</tr>
<tr>
<td>Diagnostic</td>
<td>1 1 1 1 0 0 0 1</td>
</tr>
<tr>
<td><strong>Registration</strong></td>
<td></td>
</tr>
<tr>
<td>Registration</td>
<td>1 1 1 1 0 0 1 1</td>
</tr>
<tr>
<td>Registration confirmation</td>
<td>1 1 1 1 0 1 1 1</td>
</tr>
</tbody>
</table>

Table B-2 Packet Type Identifier

For Modulo 8, bits 6, 7, and 8 in RR, RNR, REJ, or data packets contain the packet receive sequence number P(R). Bits 2, 3, and 4 in data packets contain the packet send sequence number P(S). Bit 5 in data packets contains the M (more) bit. If this bit is set to 1, more data is to follow.
B.4 Example of Call Setup, Data Transfer, and Call Clear

Figure B-1 is an example of the commands and expected responses for call setup, data transfer, and call clear for an SVC.

Figure B-1 Call Setup, Data Transfer, and Call Clearing
The XID procedures are implemented in accordance with CCITT Recommendation Q.921 (COM XI–R 12–E, DEC 1985), APPENDIX IV. However, there are some shortcomings in this description:

There is no description of the ‘negotiation’ aspect of the procedure – i.e. what does a layer 2 do with the XID parameters that it receives in an XID frame? The approach used in this implementation is:

- If an XID command is received, the layer 2 compares the received values with its internal preferred values. For N201–TX, N201–RX, and K, it adopts the lower values for its subsequent user; and for T200 it adopts the higher value for its subsequent user.
- The resulting XID response frame that it sends contains the new layer 2 values – i.e. the values just adopted.
- If an XID response is received, the layer 2 adopts the values exactly as received (except for the K value) for its subsequent use, since the side that sent the response (as a result of having received an XID command) has already adopted values that are compatible with both sides.

There is no provision for both sides of the data link to negotiate different window sizes (only K–TX is allowed in the XID message). The approach used in this implementation is to allow both sides to have their own unique K values:

- The layer 2 really wants the peer layer 2 to adopt a K that will keep the peer from sending too many I frames (which may overflow the receive buffer causing this layer 2 to lose some frames). The objective is to reduce, or prevent reject situations. To achieve this, the value placed in the K–TX parameter field in the outgoing XID frame is actually what it wishes the peer side to use for its K value – in this case, the K–RX variable which reflects the size of the receive buffer. Note that the K value that this side wants to use is not sent.
- When a frame is received, the layer 2 adopts the lower of the received K value and its own K value.

The definition of what constitutes an XID frame with a ‘zero I field’ can be interpreted 2 ways: there are either 3 bytes (the frame header only) or there are 7 bytes (frame header plus GI, plus FI, plus a 2 byte length field – which equals 0). The approach used in this implementation is:

- Either type of ‘short’ frame will be accepted. If it is a command, a full length XID response frame (21 bytes) is returned.
- When an XID command is sent via the state machine, a full length frame is always sent.
- Short XID frames of either type, both responses and commands, can be sent outside of the state machine.

No options are described for how many of the 4 parameters must be present in the XID frame. Hence, any XID frame that is not 3, 7, or 21 bytes long is considered to be an incorrect length.

The procedure is not described if an XID frame has 1 or more incorrect control codes. Hence, the frame is considered to be totally invalid if any incorrect code is detected.
The procedure is not described if an XID frame is received with either P or F=1. This implementation treats it as an error condition by notifying the ML or ASP, via the MDL-ERROR-INDICATION primitive (ERROR CODE=8), and then discards it.

The procedure is not described if an unsolicited XID response frame (F=0) is received, i.e. when this side has not sent an XID command (P=0) earlier. This implementation simply ignores (discards) it.

The procedure concerning how the network changes its T201 duration when different T200 value are acquired for the various data links on one basic access is not described. In this network emulation implementation, T201 is only used when the network initiates a 'link audit', i.e. by sending out the ID CHECK REQUEST message (Al= 127). Hence, only one value of T201 is used to time the responses to this request. To accommodate the slowest terminal (as indicated by the duration of its T200), T201 is made equal to the longest T200 duration of any of the terminals that have TEI's assigned to them. Thus, part of the XID procedure includes comparing the T200 just negotiated with the current T201 value, and making T201 equal to the longer of these durations. (Although this is also done on the user emulation as well, T201 is not used, so its value is irrelevant.)

The procedures concerning how a terminal changes its T202 duration are not described. In this user implementation, T202 is made equal to twice the value of the T200 duration just negotiated. In actual fact, this has no real impact because T202 is only used by the user emulation while the TEI is being acquired – i.e. before the XID procedure is executed, and a possibly different value of T200 acquired. Hence, during the time that T202 is used, the default value (2 seconds) would actually be used. (Although this is also done for the network emulation as well, T202 is not used, so its value is irrelevant). If, however, the user is caused to request a new TEI without the user emulation being re-initialized, the new request would be governed by the T202 duration negotiated in the previous session.
This section outlines some coding and style conventions recommended by IDACOM. Although you can develop your own style, it is suggested to stay close to these standards to enhance readability.

**D.1 Stack Comments**

A stack comment is surrounded by parentheses, and shows two stack pictures. The first picture shows any items or 'input parameters' that are consumed by the command; the second picture shows any items or 'output parameters' returned by the command.

Example:
The '=' command has the following stack comment.

```
( n1 \n2 -- flag )
```

In this example, n1 and n2 are numbers and the flag is either 0 for a false result, or 1 for a true result. This same example could also be written as follows.

```
( n1 \n2 -- 0|1 )
```

The '\' character separates parameters when there is more than one. The parameters are listed from left to right with the leftmost item representing the bottom of the stack and the rightmost item representing the top of the stack.

The '|' character indicates that there is more than one possible output. The above example indicates that either a 0 (false result) or a 1 (true result) is returned on the stack after the '=' operation.
D.2 Stack Comment Abbreviations

Following is a list of commonly used abbreviations. In most cases, the stack comments shown in this manual have been written in full rather than abbreviated.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Memory address</td>
</tr>
<tr>
<td>b</td>
<td>8 bit byte</td>
</tr>
<tr>
<td>c</td>
<td>7 bit ASCII character</td>
</tr>
<tr>
<td>n</td>
<td>16 bit signed integer</td>
</tr>
<tr>
<td>d</td>
<td>32 bit signed integer</td>
</tr>
<tr>
<td>u</td>
<td>32 bit unsigned integer</td>
</tr>
<tr>
<td>f</td>
<td>Boolean flag (0=false, non-zero=true)</td>
</tr>
<tr>
<td>ff</td>
<td>Boolean false flag (zero)</td>
</tr>
<tr>
<td>tf</td>
<td>Boolean true flag (non-zero)</td>
</tr>
<tr>
<td>s</td>
<td>String (actual address of a character string which is stored in a count prefixed manner)</td>
</tr>
</tbody>
</table>

Table D-1 ITL Symbols

D.3 Program Comments

Program comments appear in source code surrounded by parentheses. These describe the intent or purpose of the definition or line of code.

There must be at least one space on each side of the parentheses.

Example:

```plaintext
HELLO ( -­ HELLO•
" HELLO"
W.NOTICE
)
```

( Display text Hello in Notice Window )
( Create string )
( Output to Notice Window )

The program comment should be kept to a minimum and yet contain enough information that another programmer can tell the intent at a glance.

D.4 Test Manager Constructs

Coding conventions for user test scripts should generally follow the style presented throughout this manual.

Indenting nested program structures should be done using the tab key in the editor. The use of many meaningful comments is highly recommended to enhance the continued maintainability of the program.
Example:
(State definition purpose comment)

0 STATE{
    EVENT Recognition Commands ( Comment )
    ACTION{
        Action Commands ( Comment )
        IF ( Comment )
        ... ( Comment )
        ... ( Comment )
        ENDIF
    }ACTION
}STATE

**D.5 Spacing and Indentation Guidelines**

The following list outlines the general guidelines for spacing and indentations:

- One space between colon and name in colon definitions.
- One space between opening parenthesis and text in comments.
- One space between numbers and words within a definition.
- One space between initial " in strings (i.e. with " string", W." string", T." string", P." string", X" hex characters", etc...)
- One or more spaces at the end of each line unless defining a string which requires additional characters.
- Tab for nested constructs.
- Carriage return after colon definition and stack comment.
- Carriage return after last line of code in colon definition and semi-colon.

See the examples in Appendices A.6 and A.4.
D.6 Colon Definitions

The colon definition should be preceded by a short comment and should start at the first column of a line. All codes underneath the definition name should be preceded by one tab. Each element within the colon definition should be well defined.

Example:

( Description of command )

: COMMANDNAME
    .....  
    IF
        .....  
        DOCASE
        CASE X [ ... ]  
        CASE Y [ ... ]  
        CASE DUP [ ... ]
        ENDCASE
    ELSE
        BEGIN
            .....  
            .....  
            UNTIL
        ENDIF
<table>
<thead>
<tr>
<th>HEX</th>
<th>DEC</th>
<th>OCT</th>
<th>ASCII</th>
<th>EBCDIC</th>
<th>HEX</th>
<th>DEC</th>
<th>OCT</th>
<th>ASCII</th>
<th>EBCDIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>00</td>
<td>NUL</td>
<td>NUL</td>
<td>30</td>
<td>48</td>
<td>60</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>01</td>
<td>SOH</td>
<td>SOH</td>
<td>31</td>
<td>49</td>
<td>61</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>02</td>
<td>STX</td>
<td>STX</td>
<td>32</td>
<td>50</td>
<td>62</td>
<td>2</td>
<td>SYN</td>
</tr>
<tr>
<td>03</td>
<td>3</td>
<td>03</td>
<td>ETX</td>
<td>ETX</td>
<td>33</td>
<td>51</td>
<td>63</td>
<td>3</td>
<td>IR</td>
</tr>
<tr>
<td>04</td>
<td>4</td>
<td>04</td>
<td>EOT</td>
<td>PF</td>
<td>34</td>
<td>52</td>
<td>64</td>
<td>4</td>
<td>PP</td>
</tr>
<tr>
<td>05</td>
<td>5</td>
<td>05</td>
<td>ENQ</td>
<td>HT</td>
<td>35</td>
<td>53</td>
<td>65</td>
<td>5</td>
<td>TRN</td>
</tr>
<tr>
<td>06</td>
<td>6</td>
<td>06</td>
<td>ACK</td>
<td>LC</td>
<td>36</td>
<td>54</td>
<td>66</td>
<td>6</td>
<td>NBS</td>
</tr>
<tr>
<td>07</td>
<td>7</td>
<td>07</td>
<td>BEL</td>
<td>DEL</td>
<td>37</td>
<td>55</td>
<td>67</td>
<td>7</td>
<td>EOT</td>
</tr>
<tr>
<td>08</td>
<td>8</td>
<td>08</td>
<td>BS</td>
<td>GE</td>
<td>38</td>
<td>56</td>
<td>70</td>
<td>8</td>
<td>SBS</td>
</tr>
<tr>
<td>09</td>
<td>9</td>
<td>09</td>
<td>HT</td>
<td>SPS</td>
<td>39</td>
<td>57</td>
<td>71</td>
<td>9</td>
<td>IT</td>
</tr>
<tr>
<td>0A</td>
<td>10</td>
<td>0A</td>
<td>LF</td>
<td>RPT</td>
<td>3A</td>
<td>58</td>
<td>72</td>
<td>:)</td>
<td>RFF</td>
</tr>
<tr>
<td>0B</td>
<td>11</td>
<td>0B</td>
<td>VT</td>
<td>VT</td>
<td>3B</td>
<td>59</td>
<td>73</td>
<td>:)</td>
<td>CU3</td>
</tr>
<tr>
<td>0C</td>
<td>12</td>
<td>0C</td>
<td>FF</td>
<td>FF</td>
<td>3C</td>
<td>60</td>
<td>74</td>
<td>&lt;</td>
<td>DC4</td>
</tr>
<tr>
<td>0D</td>
<td>13</td>
<td>0D</td>
<td>CR</td>
<td>CR</td>
<td>3D</td>
<td>61</td>
<td>75</td>
<td>=</td>
<td>NAK</td>
</tr>
<tr>
<td>0E</td>
<td>14</td>
<td>0E</td>
<td>SO</td>
<td>SO</td>
<td>3E</td>
<td>62</td>
<td>76</td>
<td>&gt;</td>
<td></td>
</tr>
<tr>
<td>0F</td>
<td>15</td>
<td>0F</td>
<td>SI</td>
<td>SI</td>
<td>3F</td>
<td>63</td>
<td>77</td>
<td>?</td>
<td>SUB</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>10</td>
<td>DLE</td>
<td>DLE</td>
<td>40</td>
<td>64</td>
<td>100</td>
<td>@</td>
<td>SP</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>11</td>
<td>DC1</td>
<td>DC1</td>
<td>41</td>
<td>65</td>
<td>101</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>12</td>
<td>DC2</td>
<td>DC2</td>
<td>42</td>
<td>66</td>
<td>102</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>19</td>
<td>13</td>
<td>DC3</td>
<td>DC3</td>
<td>43</td>
<td>67</td>
<td>103</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>20</td>
<td>14</td>
<td>DC4</td>
<td>RES</td>
<td>44</td>
<td>68</td>
<td>104</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>15</td>
<td>NAK</td>
<td>NL</td>
<td>45</td>
<td>69</td>
<td>105</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>22</td>
<td>16</td>
<td>SYN</td>
<td>BS</td>
<td>46</td>
<td>70</td>
<td>106</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>23</td>
<td>17</td>
<td>ETB</td>
<td>POC</td>
<td>47</td>
<td>71</td>
<td>107</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>24</td>
<td>18</td>
<td>CAN</td>
<td>CAN</td>
<td>48</td>
<td>72</td>
<td>110</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>25</td>
<td>19</td>
<td>EM</td>
<td>EM</td>
<td>49</td>
<td>73</td>
<td>111</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>1A</td>
<td>26</td>
<td>1A</td>
<td>EM</td>
<td>EM</td>
<td>4A</td>
<td>74</td>
<td>112</td>
<td>J</td>
<td>cent</td>
</tr>
<tr>
<td>1B</td>
<td>27</td>
<td>1B</td>
<td>ESC</td>
<td>CUI</td>
<td>4B</td>
<td>75</td>
<td>113</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>1C</td>
<td>28</td>
<td>1C</td>
<td>FS</td>
<td>IFS</td>
<td>4C</td>
<td>76</td>
<td>114</td>
<td>L</td>
<td>&lt;</td>
</tr>
<tr>
<td>1D</td>
<td>29</td>
<td>1D</td>
<td>GS</td>
<td>IGS</td>
<td>4D</td>
<td>77</td>
<td>115</td>
<td>M</td>
<td>(</td>
</tr>
<tr>
<td>1E</td>
<td>30</td>
<td>1E</td>
<td>RS</td>
<td>IRS</td>
<td>4E</td>
<td>78</td>
<td>116</td>
<td>N</td>
<td>+</td>
</tr>
<tr>
<td>1F</td>
<td>31</td>
<td>1F</td>
<td>US</td>
<td>IU5</td>
<td>4F</td>
<td>79</td>
<td>117</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td>20</td>
<td>SP</td>
<td>DS</td>
<td>50</td>
<td>80</td>
<td>120</td>
<td>P</td>
<td>&amp;</td>
</tr>
<tr>
<td>21</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>SOS</td>
<td>51</td>
<td>81</td>
<td>121</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>FS</td>
<td>52</td>
<td>82</td>
<td>122</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>WUS</td>
<td>53</td>
<td>83</td>
<td>123</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>BYP</td>
<td>54</td>
<td>84</td>
<td>124</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>LF</td>
<td>55</td>
<td>85</td>
<td>125</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>ETB</td>
<td>56</td>
<td>86</td>
<td>126</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>39</td>
<td>27</td>
<td>'</td>
<td>ESC</td>
<td>57</td>
<td>87</td>
<td>127</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>SA</td>
<td>58</td>
<td>88</td>
<td>130</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>SFE</td>
<td>59</td>
<td>89</td>
<td>131</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>2A</td>
<td>42</td>
<td>2A</td>
<td>&quot;</td>
<td>SM/SW</td>
<td>5A</td>
<td>90</td>
<td>132</td>
<td>Z</td>
<td>!</td>
</tr>
<tr>
<td>2B</td>
<td>43</td>
<td>2B</td>
<td>+</td>
<td>CSP</td>
<td>5B</td>
<td>91</td>
<td>133</td>
<td>[</td>
<td>$</td>
</tr>
<tr>
<td>2C</td>
<td>44</td>
<td>2C</td>
<td>.</td>
<td>MFA</td>
<td>5C</td>
<td>92</td>
<td>134</td>
<td>\</td>
<td></td>
</tr>
<tr>
<td>2D</td>
<td>45</td>
<td>2D</td>
<td>-</td>
<td>ENQ</td>
<td>5D</td>
<td>93</td>
<td>135</td>
<td>)</td>
<td></td>
</tr>
<tr>
<td>2E</td>
<td>46</td>
<td>2E</td>
<td>.</td>
<td>ACK</td>
<td>5E</td>
<td>94</td>
<td>136</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>BEL</td>
<td>5F</td>
<td>95</td>
<td>137</td>
<td>_</td>
<td></td>
</tr>
<tr>
<td>HEX</td>
<td>DEC</td>
<td>OCT</td>
<td>ASCII</td>
<td>EBCDIC</td>
<td>HEX</td>
<td>DEC</td>
<td>OCT</td>
<td>ASCII</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>--------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>60</td>
<td>96</td>
<td>140</td>
<td></td>
<td></td>
<td>90</td>
<td>144</td>
<td>220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>97</td>
<td>141</td>
<td>a</td>
<td>/</td>
<td>91</td>
<td>145</td>
<td>221</td>
<td>j</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>98</td>
<td>142</td>
<td>b</td>
<td></td>
<td>92</td>
<td>146</td>
<td>222</td>
<td>k</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>99</td>
<td>143</td>
<td>c</td>
<td></td>
<td>93</td>
<td>147</td>
<td>223</td>
<td>l</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>100</td>
<td>144</td>
<td>d</td>
<td></td>
<td>94</td>
<td>148</td>
<td>224</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>101</td>
<td>145</td>
<td>e</td>
<td></td>
<td>95</td>
<td>149</td>
<td>225</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>102</td>
<td>146</td>
<td>f</td>
<td></td>
<td>96</td>
<td>150</td>
<td>226</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>103</td>
<td>147</td>
<td>g</td>
<td></td>
<td>97</td>
<td>151</td>
<td>227</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>104</td>
<td>148</td>
<td>h</td>
<td></td>
<td>98</td>
<td>152</td>
<td>230</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>105</td>
<td>151</td>
<td>i</td>
<td></td>
<td>99</td>
<td>153</td>
<td>231</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>6A</td>
<td>106</td>
<td>152</td>
<td>j</td>
<td></td>
<td></td>
<td>9A</td>
<td>154</td>
<td>232</td>
<td></td>
</tr>
<tr>
<td>6B</td>
<td>107</td>
<td>153</td>
<td>k</td>
<td></td>
<td></td>
<td>9B</td>
<td>155</td>
<td>233</td>
<td>}</td>
</tr>
<tr>
<td>6C</td>
<td>108</td>
<td>154</td>
<td>l</td>
<td>%</td>
<td>9C</td>
<td>156</td>
<td>234</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td>6D</td>
<td>109</td>
<td>155</td>
<td>m</td>
<td></td>
<td></td>
<td>9D</td>
<td>157</td>
<td>235</td>
<td>)</td>
</tr>
<tr>
<td>6E</td>
<td>110</td>
<td>156</td>
<td>n</td>
<td>▲</td>
<td>9E</td>
<td>158</td>
<td>236</td>
<td>±</td>
<td></td>
</tr>
<tr>
<td>6F</td>
<td>111</td>
<td>157</td>
<td>o</td>
<td>?</td>
<td>9F</td>
<td>159</td>
<td>237</td>
<td>■</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>112</td>
<td>160</td>
<td>p</td>
<td></td>
<td>A0</td>
<td>160</td>
<td>240</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>113</td>
<td>161</td>
<td>q</td>
<td></td>
<td>A1</td>
<td>161</td>
<td>241</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>114</td>
<td>162</td>
<td>r</td>
<td></td>
<td>A2</td>
<td>162</td>
<td>242</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>115</td>
<td>163</td>
<td>s</td>
<td></td>
<td>A3</td>
<td>163</td>
<td>243</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>116</td>
<td>164</td>
<td>t</td>
<td></td>
<td>A4</td>
<td>164</td>
<td>244</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>117</td>
<td>165</td>
<td>u</td>
<td></td>
<td>A5</td>
<td>165</td>
<td>245</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>118</td>
<td>166</td>
<td>v</td>
<td></td>
<td>A6</td>
<td>166</td>
<td>246</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>119</td>
<td>167</td>
<td>w</td>
<td></td>
<td>A7</td>
<td>167</td>
<td>247</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>120</td>
<td>170</td>
<td>x</td>
<td></td>
<td>A8</td>
<td>168</td>
<td>250</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>121</td>
<td>171</td>
<td>y</td>
<td>\</td>
<td>A9</td>
<td>169</td>
<td>251</td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>7A</td>
<td>122</td>
<td>172</td>
<td>z</td>
<td>:</td>
<td>AA</td>
<td>170</td>
<td>252</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7B</td>
<td>123</td>
<td>173</td>
<td>{</td>
<td>#</td>
<td>AB</td>
<td>171</td>
<td>253</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>7C</td>
<td>124</td>
<td>174</td>
<td></td>
<td></td>
<td>AC</td>
<td>172</td>
<td>254</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>7D</td>
<td>125</td>
<td>175</td>
<td>}</td>
<td></td>
<td>AD</td>
<td>173</td>
<td>255</td>
<td>[</td>
<td></td>
</tr>
<tr>
<td>7E</td>
<td>126</td>
<td>176</td>
<td>=</td>
<td></td>
<td>AE</td>
<td>174</td>
<td>256</td>
<td>&gt;</td>
<td></td>
</tr>
<tr>
<td>7F</td>
<td>127</td>
<td>177</td>
<td>DEL</td>
<td>&quot;</td>
<td>AF</td>
<td>175</td>
<td>257</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>128</td>
<td>200</td>
<td>a</td>
<td></td>
<td>B0</td>
<td>176</td>
<td>260</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>129</td>
<td>201</td>
<td>b</td>
<td></td>
<td>B1</td>
<td>177</td>
<td>261</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>130</td>
<td>202</td>
<td>c</td>
<td></td>
<td>B2</td>
<td>178</td>
<td>262</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>131</td>
<td>203</td>
<td>d</td>
<td></td>
<td>B3</td>
<td>179</td>
<td>263</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>132</td>
<td>204</td>
<td>e</td>
<td></td>
<td>B4</td>
<td>180</td>
<td>264</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>133</td>
<td>205</td>
<td>f</td>
<td></td>
<td>B5</td>
<td>181</td>
<td>265</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>134</td>
<td>206</td>
<td>g</td>
<td></td>
<td>B6</td>
<td>182</td>
<td>266</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>135</td>
<td>207</td>
<td>h</td>
<td></td>
<td>B7</td>
<td>183</td>
<td>267</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>136</td>
<td>210</td>
<td>i</td>
<td></td>
<td>B8</td>
<td>184</td>
<td>270</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>137</td>
<td>211</td>
<td>j</td>
<td></td>
<td>B9</td>
<td>185</td>
<td>271</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>8A</td>
<td>138</td>
<td>212</td>
<td>k</td>
<td></td>
<td>BA</td>
<td>186</td>
<td>272</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8B</td>
<td>139</td>
<td>213</td>
<td>l</td>
<td></td>
<td>BB</td>
<td>187</td>
<td>273</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8C</td>
<td>140</td>
<td>214</td>
<td>m</td>
<td></td>
<td>BC</td>
<td>188</td>
<td>274</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8D</td>
<td>141</td>
<td>215</td>
<td>n</td>
<td></td>
<td>BD</td>
<td>189</td>
<td>275</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8E</td>
<td>142</td>
<td>216</td>
<td>o</td>
<td></td>
<td>BE</td>
<td>190</td>
<td>276</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8F</td>
<td>143</td>
<td>217</td>
<td>p</td>
<td></td>
<td>BF</td>
<td>191</td>
<td>277</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### ASCII/EBCDIC/HEX Conversion Table

<table>
<thead>
<tr>
<th>HEX</th>
<th>DEC</th>
<th>OCT</th>
<th>ASCII</th>
<th>EBCDIC</th>
<th>HEX</th>
<th>DEC</th>
<th>OCT</th>
<th>ASCII</th>
<th>EBCDIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>192</td>
<td>300</td>
<td>{</td>
<td></td>
<td>F0</td>
<td>240</td>
<td>360</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>193</td>
<td>301</td>
<td>A</td>
<td></td>
<td>F1</td>
<td>241</td>
<td>361</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>194</td>
<td>302</td>
<td>B</td>
<td></td>
<td>F2</td>
<td>242</td>
<td>362</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>195</td>
<td>303</td>
<td>C</td>
<td></td>
<td>F4</td>
<td>244</td>
<td>364</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>196</td>
<td>304</td>
<td>D</td>
<td></td>
<td>F3</td>
<td>243</td>
<td>363</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>197</td>
<td>305</td>
<td>E</td>
<td></td>
<td>F5</td>
<td>245</td>
<td>365</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>198</td>
<td>306</td>
<td>F</td>
<td></td>
<td>F6</td>
<td>246</td>
<td>366</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>199</td>
<td>307</td>
<td>G</td>
<td></td>
<td>F7</td>
<td>247</td>
<td>367</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>C8</td>
<td>200</td>
<td>310</td>
<td>H</td>
<td></td>
<td>F8</td>
<td>248</td>
<td>370</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>C9</td>
<td>201</td>
<td>311</td>
<td>I</td>
<td></td>
<td>F9</td>
<td>249</td>
<td>371</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td>202</td>
<td>312</td>
<td></td>
<td></td>
<td>FA</td>
<td>250</td>
<td>372</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CB</td>
<td>203</td>
<td>313</td>
<td></td>
<td></td>
<td>FB</td>
<td>251</td>
<td>373</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td>204</td>
<td>314</td>
<td></td>
<td></td>
<td>FC</td>
<td>252</td>
<td>374</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>205</td>
<td>315</td>
<td></td>
<td></td>
<td>FD</td>
<td>253</td>
<td>375</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE</td>
<td>206</td>
<td>316</td>
<td></td>
<td></td>
<td>FE</td>
<td>254</td>
<td>376</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF</td>
<td>207</td>
<td>317</td>
<td></td>
<td></td>
<td>FF</td>
<td>255</td>
<td>377</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>208</td>
<td>320</td>
<td>}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>209</td>
<td>321</td>
<td>J</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>210</td>
<td>322</td>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>211</td>
<td>323</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>212</td>
<td>324</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>213</td>
<td>325</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td>214</td>
<td>326</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>215</td>
<td>327</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D8</td>
<td>216</td>
<td>330</td>
<td>Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D9</td>
<td>217</td>
<td>331</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DA</td>
<td>218</td>
<td>332</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td>219</td>
<td>333</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DC</td>
<td>220</td>
<td>334</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD</td>
<td>221</td>
<td>335</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DE</td>
<td>222</td>
<td>336</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DF</td>
<td>223</td>
<td>337</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>224</td>
<td>340</td>
<td>\</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>225</td>
<td>341</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>226</td>
<td>342</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td>227</td>
<td>343</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td>228</td>
<td>344</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E5</td>
<td>229</td>
<td>345</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E6</td>
<td>230</td>
<td>346</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E7</td>
<td>231</td>
<td>347</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E8</td>
<td>232</td>
<td>350</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E9</td>
<td>233</td>
<td>351</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EA</td>
<td>234</td>
<td>352</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EB</td>
<td>235</td>
<td>353</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC</td>
<td>236</td>
<td>354</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ED</td>
<td>237</td>
<td>355</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EE</td>
<td>238</td>
<td>356</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EF</td>
<td>239</td>
<td>357</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This appendix cross references old commands, not appearing in this manual, with new replacement commands.

<table>
<thead>
<tr>
<th>Old Command</th>
<th>New Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>#INCOMING</td>
<td>#DEST</td>
</tr>
<tr>
<td>#OUTGOING</td>
<td>#ORIG</td>
</tr>
<tr>
<td>$MSG-CALLREF</td>
<td>$MSG-CRVALUE</td>
</tr>
<tr>
<td>$MSG-SOURCE</td>
<td>$MSG-CRFLAG</td>
</tr>
<tr>
<td>$MSG_ERROR</td>
<td>$MSG-ERROR</td>
</tr>
<tr>
<td>=CN_CALLED_NUM</td>
<td>*COD -&gt; CLDN_NUMBER</td>
</tr>
<tr>
<td>=CN_CALLED_SAD</td>
<td>*COD -&gt; CLDS_ADDRESS</td>
</tr>
<tr>
<td>=CN_CALLING_NUM</td>
<td>*COD -&gt; CLGN_NUMBER</td>
</tr>
<tr>
<td>=CN_CALLING_SAD</td>
<td>*COD -&gt; CLGS_ADDRESS</td>
</tr>
<tr>
<td>?CN_CALLED_NUM</td>
<td>*COD -&gt; CLDN_NUMBER</td>
</tr>
<tr>
<td>?CN_CALLED_SAD</td>
<td>*COD -&gt; CLDS_ADDRESS</td>
</tr>
<tr>
<td>?CN_CALLING_NUM</td>
<td>*COD -&gt; CLGN_NUMBER</td>
</tr>
<tr>
<td>?CN_CALLING_SAD</td>
<td>*COD -&gt; CLGS_ADDRESS</td>
</tr>
<tr>
<td>?RX_MSG</td>
<td>?L3_MSG</td>
</tr>
<tr>
<td>ADDR1*</td>
<td>MADDR1*</td>
</tr>
<tr>
<td>ADDR2*</td>
<td>MADDR2*</td>
</tr>
<tr>
<td>CMND/RESP*</td>
<td>MC/R-BIT*</td>
</tr>
<tr>
<td>CONTROL1*</td>
<td>MCONTROL1*</td>
</tr>
<tr>
<td>CONTROL2*</td>
<td>MCONTROL2*</td>
</tr>
<tr>
<td>DATA-STATUS</td>
<td>STATUS_ERR?</td>
</tr>
<tr>
<td>DL_DATA_MSG</td>
<td>MESG SEND_I</td>
</tr>
<tr>
<td>DL_DATA_POOL</td>
<td>POOL SEND_I</td>
</tr>
<tr>
<td>DL_DATA_RAW</td>
<td>RAW SEND_I</td>
</tr>
<tr>
<td>DL_DATA_REQ</td>
<td>SEND_I</td>
</tr>
<tr>
<td>DL_UDATA_MSG</td>
<td>MESG SEND_UI</td>
</tr>
<tr>
<td>DL_UDATA_POOL</td>
<td>POOL SEND_UI</td>
</tr>
<tr>
<td>DL_UDATA_RAW</td>
<td>RAW SEND_UI</td>
</tr>
<tr>
<td>DL_UNIT_DATA</td>
<td>DL_UDATA</td>
</tr>
<tr>
<td>DL_UNIT_DATA_REQ</td>
<td>SEND_UI</td>
</tr>
<tr>
<td>K-XID*</td>
<td>XID-K*</td>
</tr>
<tr>
<td>MDL_UDATA_MSG</td>
<td>MESG SEND_MUI</td>
</tr>
<tr>
<td>MDL_UDATA_POOL</td>
<td>POOL SEND_MUI</td>
</tr>
<tr>
<td>MDL_UDATA_RAW</td>
<td>RAW SEND_MUI</td>
</tr>
<tr>
<td>MDL_UNIT_DATA</td>
<td>MDL_UDATA</td>
</tr>
<tr>
<td>Old Command</td>
<td>New Command</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>M_ADDR1</td>
<td>MADDR1*</td>
</tr>
<tr>
<td>M_ADDR2</td>
<td>MADDR2*</td>
</tr>
<tr>
<td>M_CMND/RESP</td>
<td>MC/R-BIT*</td>
</tr>
<tr>
<td>M_CONTROL1</td>
<td>MCONTROL1*</td>
</tr>
<tr>
<td>M_CONTROL2</td>
<td>MCONTROL2*</td>
</tr>
<tr>
<td>M_L2-LENGTH</td>
<td>L2-LENGTH</td>
</tr>
<tr>
<td>M_L2-POINTER</td>
<td>L2-POINTER</td>
</tr>
<tr>
<td>M_L3-LENGTH</td>
<td>L3-LENGTH</td>
</tr>
<tr>
<td>M_L3-POINTER</td>
<td>L3-POINTER</td>
</tr>
<tr>
<td>M_NR</td>
<td>MNR*</td>
</tr>
<tr>
<td>M_NS</td>
<td>MNS*</td>
</tr>
<tr>
<td>M_P/F*RCVD</td>
<td>MP/F-BIT*</td>
</tr>
<tr>
<td>M_SAPI</td>
<td>MSAPI*</td>
</tr>
<tr>
<td>N201-RX-XID*</td>
<td>XID-RX-N201*</td>
</tr>
<tr>
<td>N201-TX-XID*</td>
<td>XID-TX-N201*</td>
</tr>
<tr>
<td>NO_IE_ERROR#</td>
<td>NO_IE-ERROR#</td>
</tr>
<tr>
<td>NR*</td>
<td>MNR*</td>
</tr>
<tr>
<td>NS*</td>
<td>MNS*</td>
</tr>
<tr>
<td>P/F*RCVD</td>
<td>MP/F-BIT*</td>
</tr>
<tr>
<td>PARAM_WRONG_LEN#</td>
<td>PARAM-WRONG-LEN#</td>
</tr>
<tr>
<td>R-XIDC#</td>
<td>R#XIDC</td>
</tr>
<tr>
<td>R-XID#</td>
<td>R#XID</td>
</tr>
<tr>
<td>REC-LENGTH*</td>
<td>L2-LENGTH</td>
</tr>
<tr>
<td>REC-POINTER*</td>
<td>L2-POINTER</td>
</tr>
<tr>
<td>S:L_MSG</td>
<td>MESG S:L</td>
</tr>
<tr>
<td>S:L_POOL</td>
<td>POOL S:L</td>
</tr>
<tr>
<td>S:L_RAW</td>
<td>RAW S:L</td>
</tr>
<tr>
<td>S:MSG</td>
<td>MESG SEND_I</td>
</tr>
<tr>
<td>S:MUL_MSG</td>
<td>MESG S:MUI</td>
</tr>
<tr>
<td>S:MUL_POOL</td>
<td>POOL S:MUI</td>
</tr>
<tr>
<td>S:MUL_RAW</td>
<td>RAW S:MUI</td>
</tr>
<tr>
<td>S:UL_MSG</td>
<td>MESG S:UL</td>
</tr>
<tr>
<td>S:UL_POOL</td>
<td>POOL S:UL</td>
</tr>
<tr>
<td>S:UL_RAW</td>
<td>RAW S:UL</td>
</tr>
<tr>
<td>SENDP</td>
<td>DL_DATA</td>
</tr>
<tr>
<td>SEND_L_MSG</td>
<td>MESG SEND_L</td>
</tr>
<tr>
<td>SEND_L_POOL</td>
<td>POOL SEND_L</td>
</tr>
<tr>
<td>SEND_L_RAW</td>
<td>RAW SEND_L</td>
</tr>
<tr>
<td>SEND_MESSAGE</td>
<td>POOL SEND</td>
</tr>
<tr>
<td>SEND_MESSAGE/CR</td>
<td>POOL SEND</td>
</tr>
<tr>
<td>SEND_MUL_MSG</td>
<td>MESG SEND_MUI</td>
</tr>
<tr>
<td>SEND_MUL_POOL</td>
<td>POOL SEND_MUI</td>
</tr>
</tbody>
</table>

[continued]
<table>
<thead>
<tr>
<th>Old Command</th>
<th>New Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEND_MUI_RAW</td>
<td>RAW SEND_MUI</td>
</tr>
<tr>
<td>SEND_UI_MSG</td>
<td>MESG SEND_UI</td>
</tr>
<tr>
<td>SEND_UI_POOLS</td>
<td>POOL SEND_UI</td>
</tr>
<tr>
<td>SEND_UI_RAW</td>
<td>RAW SEND_UI</td>
</tr>
<tr>
<td>SEND_UNIT_MESSAGE</td>
<td>POOL SEND_I</td>
</tr>
<tr>
<td>SET_LONG n</td>
<td>n LONG-INTERVAL !</td>
</tr>
<tr>
<td>SET_SHORT n</td>
<td>n SHORT-INTERVAL !</td>
</tr>
<tr>
<td>SET_SPEED n</td>
<td>n INTERFACE_SPEED !</td>
</tr>
<tr>
<td>SU:MSG</td>
<td>MESG SEND_UI</td>
</tr>
<tr>
<td>S_DISC</td>
<td>S:DISC</td>
</tr>
<tr>
<td>S_DM</td>
<td>S:DM</td>
</tr>
<tr>
<td>S_FRMR</td>
<td>S:FRMR</td>
</tr>
<tr>
<td>S_I</td>
<td>S:DATA</td>
</tr>
<tr>
<td>S_REJ</td>
<td>S:REJ</td>
</tr>
<tr>
<td>S_REJC</td>
<td>S:REJC</td>
</tr>
<tr>
<td>S_RNR</td>
<td>S:RNR</td>
</tr>
<tr>
<td>S_RNRC</td>
<td>S:RNRC</td>
</tr>
<tr>
<td>S_RR</td>
<td>S:RR</td>
</tr>
<tr>
<td>S_RRC</td>
<td>S:RRC</td>
</tr>
<tr>
<td>S_SABM</td>
<td>S:SABM</td>
</tr>
<tr>
<td>S_SABME</td>
<td>S:SABME</td>
</tr>
<tr>
<td>S_UA</td>
<td>S:UA</td>
</tr>
<tr>
<td>S_UI</td>
<td>S:UDATA</td>
</tr>
<tr>
<td>S_XID</td>
<td>S:XID</td>
</tr>
<tr>
<td>S_XID0</td>
<td>S:XID0</td>
</tr>
<tr>
<td>S_XID4</td>
<td>S:XID4</td>
</tr>
<tr>
<td>S_XIDC</td>
<td>S:XIDC</td>
</tr>
<tr>
<td>S_XIDC0</td>
<td>S:XIDC0</td>
</tr>
<tr>
<td>S_XIDC4</td>
<td>S:XIDC4</td>
</tr>
<tr>
<td>UL3</td>
<td>DL_UDATA</td>
</tr>
<tr>
<td>UL_M</td>
<td>MDL_UDATA</td>
</tr>
<tr>
<td>VSAPI*</td>
<td>MSAPI*</td>
</tr>
</tbody>
</table>

[continued]
INDEX

?ABORT, 20-6
ACTION, 20-1
ACTIVATED?, 3-2
ACTIVATE_DISK, 7-2
ACTIVATE_RAM, 7-2
ACTIVATE_REPORT, 7-1
AI*, 10-4
-
ALL, 8-9
ALLOC_MESSAGE, 18-22
ALLOC_BUFFER, 20-16
ALL_EXCLUDED, 18-14
ALL_INCLUDED, 18-14
ALL_LICENSES, 18-14
ALL_SELECT, 18-12
ALL_UNSELECT, 18-12
ALTER_CALL_REF, 18-23
ALTER_CR, 18-7
APC, 17-6
APC*, 17-6
APPEND_TO_BUFFER, 20-16
Architecture
emulation, 15-1 to 15-4
monitor, 2-1 to 2-4
ASPST, 17-7
ATEI, 16-16
AUTO_MODE, 18-12
B, see BACKWARD
BACKWARD, 2-3
Basic Rate Access
emulation configuration, 16-2 to 16-9
monitor configuration, 3-1 to 3-3
BB, see SCRNLBACK
BCSRC, 3-2, 16-5
BCALLOC, 18-10
BCALLOC, 18-10
BC_CONN, 18-10
BC_CONN, 18-10
BCDISC, 18-10
BCFREE, 18-10
BC_INIT, 18-10
BC_SELECT, 18-10
Bit Rate, 3-5, 6-8, 16-11
Block Number
decode, 9-1, 10-1
display format, 6-3
BLOCK-COUNT, 9-1, 10-1
BLOCK_ON, 6-3

-both, 8-2
BOTTOM, 2-4
BRI, 20-3
BROADCAST Frames, 17-1
BUFFER, 20-17
Buffer(s), 20-15 to 20-17
allocating memory, 20-16
appending text, 20-16
clearing, 20-17
information element, 18-7, 18-15
message pool, 18-20 to 18-24
moving text, 20-16
number, 20-16
packet, 17-13, 17-15, 17-17
sending, 20-17, 20-18
size, 20-15
structure, 20-15
BUFFER_SEND, 20-18
BUFFER_SENDP, 20-17
Bus
activation/deactivation, 3-2, 16-9
configuration, 3-1, 16-2
power sources, 16-3
Busy, 17-10
BYTES, 18-1
C-WCALLED, 7-11
C1=ALL, 7-2
C1=NONE, 7-2
C2=ALL, 7-5
C2=NONE, 7-5
C3+, see Filters, X.25 layer 3
C3-, see Filters, X.25 layer 3
CALL, 19-29
Call Accept
address fields, 19-8
facilities, 12-4
transmitting, 19-33
Call Reference
connection management, 18-6, 18-7
decode, 11-2
pool entry, 18-22
Call Request
address fields, 12-2, 12-3, 19-8
facilities, 12-4
transmitting, 19-29, 19-32
Call User Data
call accept, 19-8
call request, 19-12
creating, 19-12
decode, 12-5, 19-24
-CALLED, 19-14
Called Address
decode, 12-2, 19-23
filters, 7-10, 7-11
setting, 19-14, 19-15
-CALLING, 19-15
Calling Address
decode, 12-3, 19-23
filters, 7-10, 7-11
setting, 19-15
Capture RAM
capturing to RAM, 4-1, 4-2
clearing, 4-2
configuring, 4-1, 4-2
playback, 2-2 to 2-4
printing, 4-4
saving to disk, 4-2 to 4-4
tigger, 8-8
-CAPT_FULL, 4-2
-CAPT_OFF, 4-2
-CAPT_WRAP, 4-1
CCIIT Message Set, 14-1 to 14-32
CES1, 16-15
CH, 19-13
CH1, 19-13
Character Set
ASCII, 6-6
EBCDIC, 6-6
hex, 6-6
ISO, 6-6
-CCLASS, 19-8
CLEAR, 19-29
Clear Request
cause byte, 12-1, 19-20, 19-21
diagnostic byte, 12-2, 19-20, 19-21
facilities, 12-4
transmitting, 19-29, 19-33
CLEARCONF, 19-18
CLEARCONF, 19-18
CLEARCONF, 19-18
CLEARREQ, 19-17, 19-18
CLEARREQ, 19-17, 19-18
CLEARALL_CNS, 18-4
CLEARALL_CNS, 18-4
CLEAR_ALL, 20-17
CLEAR_ALL, 20-17
CLEAR_CALL, 4-2
CLEAR_CALL, 4-2
CLEAR_CR, 18-7
CLEAR_CALL, 18-21
CLEAR_CALL, 18-22
Clocking, 16-11
CMND*, 17-2
CMND/RESP*, 10-2

IDACOM
ISDN Programmer's Manual

INDEX

September 1990

ISDN Programmer's Manual
INDEX [continued]

?CN, 18-3
¬CN, 18-3
CN0, 18-3
#CN0, 18-3
CN1, 18-3
¬CN1, 18-3
CN2, 18-3
¬CN2, 18-3
CN3, 18-3
¬CN3, 18-3
CN4, 18-3
¬CN4, 18-3
CN5, 18-3
¬CN5, 18-3
CN6, 18-3
¬CN6, 18-3
CN7, 18-3
¬CN7, 18-3
?CN_ALLOC, 18-4
?CN_CE, 18-6
¬CN_CE, 18-6
?CN_CHANNEL_NUM, 18-5
¬CN_CHANNEL_NUM, 18-5
?CN_CHANNEL_TYPE, 18-5
¬CN_CHANNEL_TYPE, 18-5
?CN_CR_FLAG, 18-6
¬CN_CR_FLAG, 18-6
?CN_CR_VALUE, 18-6
¬CN_CR_VALUE, 18-6
CN_DEFAULT_TIMERS, 18-9
?CN_FREE, 18-4
¬CN_IE_BUFFER, 18-7
CN_INIT, 18-4
¬CN_INIT_PD, 18-6
?CN_LAST_TIMER, 18-9
?CN_NAT_PD, 18-6
¬CN_NAT_PD, 18-6
CN_START_TIMER, 18-9
CN_STOP_ALL_TIMERS, 18-9
CN_STOP_TIMER, 18-9
?CN_TIMER, 18-9, 20-11
?CN_TIMER_DUR, 18-9
¬CN_TIMER_DUR, 18-9
?CN_TM_STATE, 18-5, 20-18
¬CN_TM_STATE, 18-5, 20-18
¬COD, 18-13
CODE_TYPE, 3-2, 16-2
COMMAND, 20-4
Command/Response Bit
decode, 10-2
setting, 17-2
test manager event, 20-4
COMPACT_POOL, 18-22
Comparison
anchored, 11-3, 20-5, 20-6, 20-9
unanchored, 20-5, 20-7
wildcard, 20-9
CONCAT_POOLS, 18-22
Configuration
capture RAM, 4-1, 4-2
emulation, 16-1 to 16-22
interface, 3-4, 16-10
layer 1, 3-1 to 3-5, 16-2 to 16-11
layer 2, 3-5, 16-12 to 16-22
monitor, 3-1 to 3-5
X.25 layer 3, 19-1 to 19-18
Connection
allocation, 18-4
data structures, 18-5, 18-6
initialization, 18-4
selection, 18-3
timers, 18-8
Connectors
V.11, 3-4, 16-10
V.28/RS-232C, 3-4, 16-10
V.35, 3-4, 16-10
V.36, 3-4, 18-10
Control Leads, 20-13
CONT_OFF, 17-17
CONT_ON, 17-17
COPY_BUFFERS, 18-6
COPY_CN, 18-4
COPY_IE, 18-7
COPY_IE_BUF, 18-8
COPY_MESSAGE, 18-22
COPY_PARAMS, 18-7
Counters
advance poll, 17-6
poll response, 17-5
retransmission, 17-5, 17-6
TEI ID check retransmission, 17-6
XID retransmission, 17-6
CRC Error(s), 20-5
CRC_ERROR, 17-18
?CRC_ERROR, 20-5
Crossloop, 17-7
?CR_LENGTH, 18-6
¬CR_LENGTH, 18-6
CR_TO_CN, 20-19
CRVAL_MAX, 18-6
CS=ASCII, 6-6
CS=EBCDIC, 6-6
CS=HEX, 6-6
CS=JIS8, 6-6
CTOD_OFF, 4-3
CTOD_ON, 4-3
CTRIG_OFF, 7-10
CTRIG_ON, 7-10
C_FILTER, 7-9
D Bit, 12-1, 19-20, 19-22
D=WCALLED, 7-11
D1=ALL, 7-3
D1=NONE, 7-3
D2=ALL, 7-5
D2=NONE, 7-5
D3+, see Filters, X.25 layer 3
D3-, see Filters, X.25 layer 3
DATA, 19-30
Data Field
comparison, 20-9
defining, 19-30, 20-18
size, 12-2, 19-3, 19-25, 19-26, 20-17
user-defined, 19-30, 19-31
Data Packet(s)
data field length, 12-2
echoing, 19-16
M bit, 12-2, 19-20, 19-22
pointer, 12-2
transmitting, 19-30, 19-34
window size, 19-15, 20-9
Data Source, 18-1
DATA_LENGTH, 12-2
DATA_POINTER, 12-2
DATACOM_SAP#, 3-5, 16-14
DATA_CHAR, 6-6
DATA_HEX, 6-6
DATA_OFF, 6-6
DCE_END, 19-1
DEACTIVATED?, 3-2
DEACTIVATE_DISK, 7-2
DEACTIVATE_RAM, 7-2
DEACTIVATE_REPORT, 7-2
*DEC, 11-7
Decode
errors, 11-2, 11-5
information elements, 11-4, 11-5
layer 1, 9-1
layer 2, 10-1 to 10-4
layer 3, 11-1 to 11-8
timer, 19-5
X.25 layer 3, 19-19 to 19-26
DEFINE_DATA, 19-30
DEFINE_IE, 18-15
INDEX [continued]

DEST_SIDE, 18-2
Determine_ON, 20-19
Diagnostic Byte, 12-2, 19-20
DISC Frame
  data link release, 17-9
  transmitting, 17-11, 17-16
DISK_FULL, 5-1
  =DISK_FULL, 8-8
  +DISK_FULL, 8-8
DISK_OFF, 5-2
DISK_WRAP, 5-1
Display Format, 6-1 to 6-9
  block number, 6-3
  character, 6-4, 6-5
  character set, 6-6
  complete, 6-3, 6-4
  dual, 6-6
  hex, 6-4, 6-5
  layer, 2, 6-3
  layer 3, 6-4 to 6-6
  message detail, 6-5
  mnemonic, 6-4, 6-5
  split, 6-2
  timestamp, 6-3
  trace statement, 6-2
  trace statements, 6-8
  X.25 layer 3, 6-6
DIS_REC, 5-2
  -DLCI, 8-5
  +DLCI, 8-5
DL_DATA, 17-13
  DL_ESTABLISH, 17-9
DL_RELEASE, 17-9
DL_UDATA, 17-14
DM Frame, 17-11, 17-16
DO_ABORT, 17-8
DROUTING, 3-3, 16-5
DTE_END, 19-1
DTRIG_OFF, 7-11
DTRIG_ON, 7-10
D_FILTER, 7-8

ECHO_CAFAC, 19-9
ECHO_OFF, 19-16
ECHO_ON, 19-16
ELEMENT> <ELEMENT, 18-14
Emulation
  architecture, 15-1 to 15-4
  automatic, 19-3, 19-26
  configuration, 16-10 to 16-22
  manual, 19-3, 19-28
  network, 16-1
  response, 19-26 to 19-35
  user, 16-1
  X.25 layer 3 configuration, 19-1 to 19-18
EMUL_OFF, 19-28
EMUL_ON, 19-28
ENB_REC, 5-2
END_TIME, 10-1
Event Recognition, 20-3 to 20-12
  abort, 20-5
  anchored comparison, 20-5, 20-6, 20-9
  command frame, 20-4
  CRC error, 20-5
  frames, 20-4, 20-8
  from network, 20-6
  layer 1 event, 20-4
  layer 2 service, 20-5
  layer 3 data, 20-6, 20-7
  packets, 20-8
  response frame, 20-4
  timers, 20-10, 20-11
  unanchored comparison, 20-5, 20-7
  X.25 layer 3, 20-7 to 20-9
EXCLUDED, 10-14
EXTERNAL_OUT, 3-3

F, see FORWARD
  F=CR, 7-8
  F+MSG, 7-8
  F=PD, 7-7
  F=CR, 7-8
  F(MSG, 7-8
  F=PD, 7-7
  FO, 17-15
  F1, 17-15
  F3=ALL, 7-8
  F3=NONE, 7-8

Facilities
  decode, 12-4, 19-24
  fast select, 19-16, 19-17
  throughput class, 19-8
  user-defined, 19-7, 19-8, 19-10
  FAST_SELECT_OFF, 19-16
  FAST_SELECT_ON, 19-17
  FAST_SELECT_RESTRICTION, 19-17
  FF, see SCRNM_FWD

Filename
  disk recording, 2-3
  playback, 2-3
  FILE>, BUFFER, 20-16

Filters
  7-1 to 7-15
  activate, 7-1
  deactivate, 7-1
  layer 1, 7-1 to 7-4
  layer 2, 7-5, 7-6
  layer 3, 7-7 to 7-9
  selective address, 7-11
  X.25 layer 3, 7-9 to 7-15

Format, see Display Format

FORWARD, 2-3
Frame Layer, see Layer 2
Frame(s)
  DISC, 17-11, 17-16
  DM, 17-11, 17-16
  format, 17-11, 17-16
  FRMR, 10-3, 17-3 to 17-5, 17-12
  identifiers, 10-2, 20-4
  information, 17-10 to 17-14, 17-16, 17-17, 17-20
  REJ, 17-11, 17-16
  RNR, 17-10, 17-11, 17-16
  RR, 17-10, 17-11, 17-16
  SABM, 17-9, 17-10, 17-15
  SABME, 17-9, 17-11, 17-15
  transmitting, 17-10 to 17-12, 17-15 to 17-17, 18-2
  UA, 17-11, 17-16
  UI, 10-4, 17-12, 17-17
  XID, 17-4, 17-9, 17-12, 17-16

FRAME-ID, 10-2
FREEZE, 2-4, 15-4
FRMR Frame
  response bits, 17-3 to 17-5
  transmitting, 17-12, 17-16
  FRMR=C/R-BIT, 10-3
  FRMR=C/NCTL1*, 10-3
  FRMR=C/NCTL2*, 10-3
  FRMR=VR1, 10-3
  FRMR=VR2, 10-3
  FRMR=W-BIT, 10-3
  FRMR=X-BIT, 10-3
  FRMR=Y-BIT, 10-3
  FRMR=Z-BIT, 10-3
  FRMRW*, 17-4
  FRMRX*, 17-4
  FRMRY*, 17-4
  FRMRZ*, 17-4
  FRM_CHAR, 6-4
  FRM_COMP, 6-4
  FRM_HEX, 6-4
  FRM_MNEM, 6-4
  FRM_OFF, 6-3
  FROM_CAPTURE, 2-2
  FROM_DISK, 2-2
  FROM_NT, 20-6
INDEX [continued]

FTEI, 16-16
FULL, 6-7
= F_CR, 7-8
= F_PD, 7-7

General Format Identifier, 12-1, 19-20, 19-23, B-2
GET_PS, 16-4
GOOD_CRC, 17-18
HALT, 2-2, 15-3
HIGHLIGHT, 8-10
HIGHLIGHT=BLUE, 8-10
HIGHLIGHT•RED, 8-10
I Frame, see Information Frame(s)
Identifiers
frame, 10-2, 20-4
decide event, 9-1, 20-4
message, 11-3
message to 11-8
packet, 20-7
IE, see Information Element(s)
IF-CLOCK, 16-11
IF•V11, 3-4, 16-10
IF•V28, 3-4, 16-10
IF•V35, 3-4, 16-10
IF•V36, 3-4, 16-10
INCLUDED, 18-14
Information Element(s)
buffers, 18-7, 18-8, 18-15
calling management, 18-7
declares commands, 11-3, 11-5
generating, 18-13 to 18-19
identifiers, 14-5, 14-6
parameters, 18-7
selecting, 18-11 to 18-13
structures, 14-6 to 14-32
Information Frame(s)
continuous transmission, 17-17
information field length (N201), 16-18
transmitting, 17-12 to 17-14
INIT_DEFAULT_TIMERS, 18-9
INIT_IE_BUFFERS, 18-7
INSERT_ENTRY, 18-22
INTERFACE, 16-10
Interface
initialization, 16-10
speed, 3-5, 6-6, 16-11
V.11, 3-4, 16-10
V.28/RS-232C, 3-4, 16-10
V.35, 3-4, 16-10
V.36, 3-4, 16-10
INTERFACE-SPEED, 3-5, 6-6, 16-11
INTERFACE_WAKEUP, 3-4, 16-10
INTERFACE_FILL, 16-11
INTERCEPT, 19-31
Interrupt User Data, 19-21, 19-22
INTERUPTED?, 3-2
Interval Timers, see Timer(s)
K, 16-19
K#, 16-20
K*, 16-19
K-CCITT, 17-7
K-RX, 16-20
KRX, 16-20
L1-ID*, 9-1
L1_ACTIVATE, 16-9
L1_COMP, 6-3
L1_DEACTIVATE, 16-9
L1_EVENT, 20-4
L1_OFF, 6-3
L2-LENGTH, 10-1
L2-POINTER, 10-1
L2_MOD128, 3-5
L2_MOD8, 3-5
L2ST, 17-7
L2_OFF, 16-14
L2_ON, 16-14
L2_SERVICE, 20-6
L3-LENGTH, 11-1
L3-POINTER, 11-1
L3_IE, 11-5
L3_IE_COUNT, 11-5
L3_IE_ERROR, 11-6
L3_IE_ORDER, 11-4
L3_MAND_ERROR, 11-5
L3_MAND_IE, 11-4
L3_MATCH, 11-3, 20-6
L3_MSG, 11-3, 20-6
L3_MSG_DIR, 11-5
L3_OFFSET, 11-6
L3_ON, 19-3
L3_OPT_ERROR, 11-5
L3_OPT_IE, 11-4
L3_SEARCH, 11-4, 20-7
L3_UNEXP_IE, 11-4
L3_VALID_MSG, 11-3
Lapse Timers, see Timer(s)
Layer 1
configuration, 3-1 to 3-5, 16-1 to 16-11
decode, 9-1
display format, 6-3
identifiers, 9-1
filters, 7-1 to 7-4
test manager events, 20-4
triggers, 8-3
Layer 2
configuration, 3-5, 16-12 to 16-22
decode, 10-1 to 10-4
description, A-7 to A-12
display format, 6-3
emulation, 17-1 to 17-18
filters, 7-5, 7-6
services, 17-9, 17-10, 20-6
test manager events, 20-4 to 20-6
triggers, 8-4 to 8-6
Layer 3, see Packet Layer
decode, 11-1 to 11-8
description, A-12 to A-15
display format, 6-4, 6-5
filters, 7-7 to 7-9
simulation, 18-1 to 18-24
test manager events, 20-6, 20-7
triggers, 8-6, 8-7
X.25, see X.25 Layer 3
LCN, 19-25, see Logical Channel(s)
= LCN, 19-13
LON, 19-13
LNCALLED, 19-15
LNCALLING, 19-15
LONCES, 19-14, 19-25
=LONCES, 19-14
LONSIZE, 19-26
LONSIZE, 19-25
LONSTATE, 19-25
LONTIMER, 20-11
LONWINDOW, 19-25
Logical Channel(s)
Description, A-7, A-8
establishment, A-8, A-9
selection, 16-15, 19-14
Live Data
capturing to RAM, 4-1
emulation, 15-2
monitor, 2-1
recording, 5-1
simulating playback, 2-4
LOAD_IE_BUFFER, 18-15
LOAD_MESSAGES, 18-21
LOAD_MESSAGE_SET, 13-2
LOAD_RETURN_STATE, 20-2
Logical Channel(s)
called address, 19-14, 19-15

ISDN Programmer's Manual
INDEX [continued]

Logical Channel(s) [continued]
calling address, 19-15
data echo, 19-16
decode, 12-1
filters, 7-12
LCN, 19-25
selection, 19-13
setup, 19-13 to 19-18
state machine, 19-25, 19-27
timer, 20-11
variables, 19-19 to 19-26
LONG-INTERVAL, 6-9
LONG_BUFFER, 18-7
M Bit, 12-2, 19-20, 19-22
M-D, 12-1
M-GFI, 12-1
M-LCB, 12-1
M-LCG, 12-1
M-LCN, 12-1
M-MORE, 12-2
M-PR, 116-2
M-PS, 116-2
M-Q, 12-1
M-RCALLED, 12-2
M-RCALLING, 12-3
M-RCAUSE, 12-1
M-RCUD, 12-5
M-RDIAG, 12-2
M-REC-PKT-ID, 12-1
M-RFAC, 12-4
MADDR1, 10-1
MADOR2", 10-1
MAKE....BUSY, 17-10
MAKE....CAFAC, 19-10
MAKE....CUD, 19-12
MAKE....FAC, 19-8
MAKE-.RFAC, 19-1
MANAGEMENT_ID•, 10-4
MAN_MODE, 18-12
MAX, 16-20
MAX_LENGTH, 20-17
MC/R-BIT 0 , 10-2
MCLR, 13-2
MCONTROL1°, 10-1
MCONTROL2°, 10-1
MD1, 17-13
MD2, 17-13
MD3, 17-13
MD4, 17-13
MD5, 17-13
MD6, 17-13
MD7, 17-13
MD8, 17-13
MDL_ASSIGN_I, 17-9
MDL_ASSIGN_R, 17-9
MDL_REMOVE, 17-10
MDL_DATA, 17-14
MSG, 18-2, 18-12
MILLISECONDS_ELAPSED, 20-14
MINUTES_ELAPSED, 20-14
MLST, 17-7
ML_OFF, 16-14
ML_ON, 16-14
MMODE-FLAG*, 3-5, 10-2
MNR, 10-2, 17-14
MNS, 10-2, 17-14
MODE, 16-16
MODE-FLAG*, 16-16
Modulo 8/128
decode, 10-2
multiple frame operation, 16-16
setting, 19-3
MODULUS*, 16-16
Monitor, 2-1, 15-2
Monitor architecture, 2-1 to 2-4
configuration, 3-1 to 3-5
live data, 2-1
MOVE_MESSAGE, 18-22
MP/F-BIT*, 10-2
MSAPI*, 10-1
MSG, 6-5
MSG+IE, 6-5
MSG+IE+PA, 6-5
$MSG-CRFLAG, 11-2
$MSG-CRLEN, 11-2
$MSG-CRVALUE, 11-2
$MSG-DISCR, 11-2
$MSG-ERROR, 11-2
$MSG-ID, 11-2
$MSG-TYPE, 11-2
MSG_CHAR, 6-5
MSG_COMP, 6-4
MSG_HEX, 6-5
MSG_MMES, 6-5
MSG_OFF, 6-4
MTEI*, 10-2
NO_CA, 19-8
NO_CAFAC, 19-9
NO_FAC, 19-7
NR, 17-14
NRVALID*, 17-7
NS, 17-14
NTWK_EMUL, 16-1
NUMINCLUDED, 18-14
Octet identifiers, 11-3 to 11-8
inclusion, 19-14
ORIG_SIDE, 19-2
OTHER_EVENT, 20-12

IDACOM
ISDN Programmer's Manual

INDEX
September 1990
INDEX [continued]

OWN_BUSY, 17-10
P0, 17-15
P1, 17-15
Packet Layer configuration, 19-2 to 19-6
modulus 8/128, 19-3
state machine, 19-2, 19-3
T10 timer, 19-5
T11 timer, 19-6
T12 timer, 19-6
T13 timer, 19-6
T20 timer, 19-4
T21 timer, 19-4
T22 timer, 19-4
T23 timer, 19-5
Packet(s)
call accept, 19-33
call request, 19-29, 19-32
call user data, 19-6, 19-12
clear confirm, 19-33
clear request, 19-29, 19-33
data, 19-30, 19-31, 19-34, 20-18
diagnostic, 19-36
identifiers, 20-7
interrupt, 19-31, 19-34
interrupt confirm, 19-34
reject, 19-35
reset confirm, 19-34
reset request, 19-31, 19-33
restart request, 19-31, 19-32
RNR, 19-35
RR, 19-34
Test manager events, 20-8
transmitting, 17-13, 19-29 to 19-35
PACKET-TYPE, 20-7
PACKET_MOD128, 19-3
PACKET_MOD8, 19-3
Parameters
connection data, 18-7
setting, 18-13, 18-14
Physical Layer, see Layer 1
PKT, 17-13
PKT?, 17-13
PKTS, 17-17
PLAYBACK, 2-3
Playback
capture RAM, 2-2
capture, 2-3, 2-4
disk recording, 2-2 to 2-4
event, 15-4
monitor, 2-1, 15-2
simultaneous live data, 2-4, 15-4
*PLAYBACK, 8-2
Poll/Final Bit
decode, 10-2
setting, 17-15
Polling, 16-13
POOL, 18-2, 18-23
Pool Entry, 18-20 to 18-24
call reference, 18-22, 18-23
comment, 18-21
file, 18-21
name, 18-20
POOL*FLAG, 18-23
*POOL*FLAG, 18-23
POOL*VALUE, 18-22
*POOL*VALUE, 18-22
PORT-ID, 9-1, 10-1
PORT_BRID, 3-1, 16-2
PORT_BRID2, 3-1, 16-2
Power Supply, 16-4
PPMP, 3-1, 16-2
PR, 19-19
PROC, 17-5
PREF-TEI, 16-17
?PRI, 20-3
Printing
capture RAM, 4-2 to 4-4
disk recording, 4-3
throughput graph, 6-9
PRINT_OFF, 4-4
PRINT_ON, 4-4
PRINT_TPR, 6-9
Protocol Discriminator
decode, 11-2
setting, 18-5
PS, 19-19
PT, 16-20
PT-N201, 16-19
PT-N201TX, 16-19
PT-T200, 16-17
PT_TO PT, 17-1
PVC, 19-14
Q Bit, 12-1, 19-20, 19-22
QUIT_TRA, 4-2
R-WCALLED, 7-11
R1=ALL, 7-2
R1=NONE, 7-2
R2=ALL, 7-5
R2=NONE, 7-5
R3+, see Filters, X25 layer 3
R3=CALL, 7-13
R3=, see Filters, X25 layer 3
R3=CALL, 7-14
R3=ALL, 7-14
R3=NONE, 7-14
RAW, 18-1
RC, 17-5
RC*, 17-5
RC-ML1*, 17-6
RC-ML2*, 17-6
RC20, 17-6
RC20*, 17-6
RC201, 17-6
RC201*, 17-6
RCALLED, 19-23
RCALLING, 19-23
RCause, 19-20
RCD, 19-20
RECALL_MESSAGE, 18-21
RECO, 19-20
RECM, 19-20
RECORD, 6-2
Recording
captured data, 4-3, 4-4
data, 2-3
disk, 5-2
overwrite, 5-1
playback, 2-2 to 2-4
stop, 5-2
suspend, 5-2
RECPKTMOD, 19-20
RECO, 19-20
RECPKT.ID, 19-20
REJ Frame, 17-11, 17-16
REJC, 17-11
Report Generator, see Display Format
REP_NORMAL, 6-2
REP_OFF, 6-2
REP_SPLIT, 6-2
REP_TRACE, 6-2
RESET, 18-31
Reset Request
cause, 12-1, 19-20, 19-21
diagnostic byte, 12-2, 19-20, 19-21
transmitting, 19-31, 19-33
RESET_ALL_IE_MAPS, 18-7
RESET_BUSY, 17-10
RESET_IE_MAP, 18-7
RESP*, 17-2
*RESPONSE, 20-4

ISDN Programmer's Manual

IDACOM
INDEX [continued]

RESTART, 19-31
Restart Request
  cause, 12-1, 19-20, 19-21
diagnostic byte, 12-2, 19-20, 19-21
transmitting, 19-31, 19-32
RETURN_STATE, 20-2
RFAC, 19-24
RI, 10-4
RIUD, 19-21
RLCN, 19-21
=RLCN1, 7-12
RLCN=OFF, 7-12
RLCN=SEL, 7-12
RLCN=ALL, 7-12
RLCN=SEL, 17-12
RNR Frame, 17-10, 17-11, 17-16
RNRC, 17-11
RR Frame, 17-10, 17-11, 17-16
RRC, 17-11
RR POLL_OFF, 16-13
RR POLL_ON, 16-13
?RR POLL_ON, 16-13
RTRIG_OFF, 7-10
RTRIG_ON, 7-10
RUN_SEQ, 20-2
?RX, 20-8
RX_CALLREF, 18-2
?RX DATA, 20-8
?RX_FRAME, 20-4
?RX MATCH, 20-5
?RX_PACKET, 20-8
?RX SEARCH, 20-5
?RX X25 DATA, 20-9
R FILTER, 7-8

S:CALLC, 19-33
S:CALLR, 19-32
S:CLEARC, 19-33
S:CLEARR, 19-33
S:DATA, 17-17
S:DATAP, 17-34
S:DIAG, 19-36
S:DISC, 17-16
S:DM, 17-16
S:FRMR, 17-16
S:I, 17-16
S:INTC, 19-34
S:INTR, 19-34
S:MUDATA, 17-17
S:MUJ, 17-17
S:REJ, 17-16
S:REJC, 17-16
S:REJP, 19-35
S:RESETC, 19-34
S:RESETR, 19-33
S:RESTARTC, 19-32
S:RESTARTR, 19-32
S:RNR, 17-16
S:RNRC, 17-16
S:RNR2, 19-35
S:RR, 17-16
S:RRC, 17-16
S:RRP, 19-34
S:SABM, 17-15
S:SABME, 17-15
S:SU, 17-16
S:SU DATA, 17-17
S:UJ, 17-17
S:XD, 17-16
S:XDO, 17-16
S:XDC, 17-16
S:XDC0, 17-16
S:XDC4, 17-16
S:A, 16-15
SABM/SABME Frames
data link establish, 17-9
transmitting, 17-10, 17-11, 17-15

SAPI, 16-16, see Service Access Point Identifier
+SAPI, 8-4
-SAPI, 8-5
+SAPI, 8-4
SAPI, 16-15
SAVE_MESSAGES, 18-21
SCAUSE, 19-21
Screen, scroll control, 2-3, 2-4
SCRN_BACK, 2-3
SCRN_FWD, 2-3
SIDIAG, 19-21
SA_DIAG-EXP, 19-22
SECCONDS_ELAPSED, 20-14
SEE_TRA, 4-3
SELECT, 18-12
SELECT CR, 18-6
Selfloop, 16-6
SEND, 19-31
SENDDB, 19-22
SENDF, 17-15
SENDGF, 19-23
SENDM, 19-22
SENDQ, 19-22
SENDJ, 17-12
SEND MUI, 17-12
SEND UI, 17-12
Sequence Numbers
decode, 10-2
emulation, 17-2, 17-3
NSU, 19-26
P(R), 12-2, 19-19, 19-26
P(S), 12-2, 19-19, 19-26
transmitting, 17-14
SEQ, 20-2
Service Access Point Identifier
decode, 10-1
description, A-7, A-8
packet data, 3-5
selection, 16-16
setting, 16-16, 18-6
triggers, 8-4
Services
description, A-6, A-7
establish, 17-9
receiver busy, 17-10
release, 17-9
send data, 17-13, 17-14
TEI assignment, 17-9
TEI removal, 17-10
test manager event, 20-6
SET FAC LEN, 19-7, 19-25
SET INTERFACE, 18-10
SET INT LEN, 19-22
SET PS, 16-4
SHORT-INTERVAL, 6-9
SIUD, 19-22
Size
data field, 19-3, 19-25, 19-26, 20-17
window, 19-25, 20-9
+SIZE, 19-3, 20-17
SM MSG, 18-17
START TIME, 9-1, 10-1
START L FASE-TIMER, 20-14
START_TIMER, 20-15
State Machine
in-state, 17-7 to 17-14
layer 2, 16-14, 17-7 to 17-9
management layer, 16-14, 17-7 to 17-9
out-of-state, 17-14 to 17-17
packet layer, 19-2, 19-3
test manager, 20-1
X25 layer, 19-27, 19-28
State Variables, 17-2, 17-3
STATE ASP*, 17-7
STATE L2*, 17-7
STATE ML*, 17-7
STATE INIT ( ) STATE INIT, 20-1
INDEX [continued]

Transmitting frames, 17-10 to 17-12, 17-15 to 17-17, 18-2 messages, 18-11, 18-17 packets, 17-13 X.25 packets, 19-29 to 19-35 TRA_ALL, 4-2 TRA_END, 4-3 TRA_START, 4-3 Triggers, 8-1 to 8-10 actions, 8-9, 8-10 arming, 8-1, 8-2 capture RAM full, 8-8 direction, 8-2 to 8-9 disk full, 8-8 layer 1, 8-3 layer 2, 8-4 to 8-6 layer 3, 8-6, 8-7 selection, 8-2 string, 8-5 time, 8-8 TX-CALLREF, 18-3 _T_CR, 8-7 _T_PD, 8-7 UA Frame, 17-11, 17-16 UI Frame: decode, 10-4 transmitting, 17-12, 17-17 UI-TYPE*, 10-4 UNSELECT, 18-13 User Data, see Call User Data USER_CAFAC, 19-9 USER_EMUL, 16-1 USER_FAC, 19-7 _USR, 8-2 USB NET, 16-1 VA, 17-2 VA*, 17-3 Voice encoding, 3-2, 16-2 tone selection, 16-3 VR, 17-3 VR*, 17-3 VS, 17-2 VS*, 17-2 VTEI*, 16-16 W, 16-20 W Bit, 10-3, 17-3 W0, 17-3 W1, 17-3 ?WAKEUP, 20-11 Window, 16-19 _WINDOW, 19-15 WINDOW?, 20-9 X Bit, 10-3, 17-4 X.25 Layer 3 automatic response, 19-28 cause byte, 12-1, 19-20, 19-21 CCITT Recommendation, B-1 configuration, 19-1 to 19-18 D bit, 12-1, 19-20, 19-22 decode, 19-19 to 19-26 filters, 7-9 to 7-15 general format Identifier, 12-1, 19-20, 19-23, B-2 logical channel, 12-1, 19-21 P(R), 12-2, 19-19 P(S), 12-2, 19-19 Q bit, 12-1, 19-20, 19-22 services, B-1 state machine, 19-27, 19-28 structure, B-1 to B-3 timers, 19-29 window, 19-25 X0, 17-4 X1, 17-4 X25_L3_OFF, 19-28 X25_L3_ON, 19-28 XID decode, 10-4 negotiate counter (NM20), 16-21 negotiate timer (TM20), 16-21 negotiation procedure, C-1, C-2 preferred window size, 16-20 timer (T200), 16-18 transmitting, 17-12, 17-16 variables, 17-6 XID-K*, 10-4 XID-RX-N201*, 10-4 XID-T200*, 10-4 XID-TX-N201*, 10-4 XIDC, 17-12 XIDV, 17-6 XID_OFF, 16-13 XID_ON, 16-13 _TXID_ON, 16-13 Y Bit, 10-3, 17-4 Y0, 17-4 Y1, 17-4 YES_CA, 19-8 YES_FAC, 19-7 Z Bit, 10-3, 17-4 Z0, 17-4 Z1, 17-4