SDLC/SNA
PROGRAMMER’S MANUAL
NOTE
SDLC Emulation:
The 3274.F test script contains character strings that are encoded in the EBCDIC character set and, therefore, cannot be edited with an ASCII editor.

1.1 Known Problem

Activation of filters with the REP_FILTER_ON/OFF commands is not required for filter commands published in the SDLC/SNA Programmer's Manual (eg. RFRM=ALL). These commands do not update the Filter Menus.
This manual is intended to provide a programmer's guide to the SNA Monitor and SDLC 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 CONFIGURATION ........................................ 2-1

3 MONITOR ARCHITECTURE ......................................... 3-1
   3.1 Live Data ............................................. 3-1
   3.2 Playback ............................................... 3-2
      Playback Control ..................................... 3-3
   3.3 Simultaneous Live Data and Playback .................... 3-4

4 MONITOR DECODE ................................................ 4-1
   4.1 Communication Variables ................................. 4-1
      Layer 1 ................................................. 4-2
      Frame (SDLC) Layer .................................... 4-3
      TH (Transmission Header) ............................. 4-4
      RH (Request/Response Header) ......................... 4-10
      SD (Sense Data) ....................................... 4-13
      RU (Request/Response Unit) .......................... 4-14

5 CAPTURE TO RAM ................................................ 5-1
   5.1 Capturing to RAM ....................................... 5-1
   5.2 Transferring from RAM .................................. 5-2
      To Disk ................................................ 5-4
      To Printer ............................................. 5-4

6 DISK RECORDING ................................................ 6-1

7 DISPLAY FORMAT ................................................ 7-1

8 FILTERS ......................................................... 8-1

9 EMULATION CONFIGURATION ...................................... 9-1
   9.1 Emulation Mode ........................................ 9-1
# TABLE OF CONTENTS (continued)

## 9 EMULATION CONFIGURATION (continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2</td>
<td>Physical Layer</td>
<td>9-2</td>
</tr>
<tr>
<td>9.3</td>
<td>Frame Layer</td>
<td>9-6</td>
</tr>
<tr>
<td>9.4</td>
<td>Station Setup</td>
<td>9-8</td>
</tr>
</tbody>
</table>

## 10 EMULATION ARCHITECTURE

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>Live Data</td>
<td>10-1</td>
</tr>
<tr>
<td>10.2</td>
<td>Playback</td>
<td>10-3</td>
</tr>
<tr>
<td>10.3</td>
<td>Simultaneous Live Data and Playback</td>
<td>10-4</td>
</tr>
</tbody>
</table>

## 11 EMULATION DECODE

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>Communications Variables</td>
<td>11-1</td>
</tr>
<tr>
<td></td>
<td>Data Link Receive</td>
<td>11-2</td>
</tr>
<tr>
<td></td>
<td>Data Link Transmit</td>
<td>11-2</td>
</tr>
<tr>
<td></td>
<td>Transmission Header Receive</td>
<td>11-3</td>
</tr>
<tr>
<td></td>
<td>Transmission Header Transmit</td>
<td>11-3</td>
</tr>
<tr>
<td></td>
<td>Request/Response Header Transmit</td>
<td>11-4</td>
</tr>
<tr>
<td></td>
<td>Sense Data Transmit</td>
<td>11-7</td>
</tr>
<tr>
<td></td>
<td>Request/Response Unit Transmit</td>
<td>11-8</td>
</tr>
</tbody>
</table>

## 12 EMULATION RESPONSE

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1</td>
<td>Emulation State Machines</td>
<td>12-1</td>
</tr>
<tr>
<td>12.2</td>
<td>Automatic Responses</td>
<td>12-3</td>
</tr>
<tr>
<td>12.3</td>
<td>State-Dependent Send Commands</td>
<td>12-3</td>
</tr>
<tr>
<td>12.4</td>
<td>State-Independent Send Commands</td>
<td>12-14</td>
</tr>
<tr>
<td>12.5</td>
<td>CRC Errors</td>
<td>12-17</td>
</tr>
</tbody>
</table>

## 13 TEST MANAGER

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.1</td>
<td>ITL Constructs</td>
<td>13-1</td>
</tr>
<tr>
<td>13.2</td>
<td>Event Recognition</td>
<td>13-2</td>
</tr>
<tr>
<td></td>
<td>Layer 1</td>
<td>13-3</td>
</tr>
<tr>
<td></td>
<td>Received Frames</td>
<td>13-3</td>
</tr>
<tr>
<td></td>
<td>Timeout Detection</td>
<td>13-14</td>
</tr>
<tr>
<td></td>
<td>Function Key Detection</td>
<td>13-15</td>
</tr>
<tr>
<td></td>
<td>Interprocessor Mail Events</td>
<td>13-15</td>
</tr>
<tr>
<td></td>
<td>Wildcard Events</td>
<td>13-16</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS (continued)

## 13 TEST MANAGER (continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.3</td>
<td>SDLC/SNA Actions</td>
<td>13-16</td>
</tr>
<tr>
<td></td>
<td>Layer 1 Actions</td>
<td>13-17</td>
</tr>
<tr>
<td></td>
<td>Protocol Actions</td>
<td>13-18</td>
</tr>
<tr>
<td></td>
<td>Using Buffers</td>
<td>13-18</td>
</tr>
</tbody>
</table>

## 14 TEST SCRIPTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.1</td>
<td>BAR_GRAPH.F</td>
<td>14-1</td>
</tr>
<tr>
<td>14.2</td>
<td>SNA_STAT.F</td>
<td>14-4</td>
</tr>
<tr>
<td>14.3</td>
<td>SNA 3274 Scenario</td>
<td>14-10</td>
</tr>
<tr>
<td></td>
<td>3274_DTE.F</td>
<td>14-18</td>
</tr>
<tr>
<td>14.4</td>
<td>Multipoint Test Scripts</td>
<td>14-21</td>
</tr>
<tr>
<td></td>
<td>PRI_MULTI.F</td>
<td>14-21</td>
</tr>
<tr>
<td></td>
<td>SEC_MULTI.F</td>
<td>14-25</td>
</tr>
<tr>
<td>14.5</td>
<td>SDLCSCRIPT.F</td>
<td>14-27</td>
</tr>
</tbody>
</table>

## APPENDICES

## A INTRODUCTION TO IBM SNA

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>SDLC Configurations</td>
<td>A-1</td>
</tr>
<tr>
<td></td>
<td>Point-to-Point</td>
<td>A-2</td>
</tr>
<tr>
<td></td>
<td>Multipoint</td>
<td>A-3</td>
</tr>
<tr>
<td>A.2</td>
<td>SNA Layers</td>
<td>A-3</td>
</tr>
<tr>
<td></td>
<td>Physical Layer</td>
<td>A-4</td>
</tr>
<tr>
<td>A.3</td>
<td>Data Link Control Layer</td>
<td>A-5</td>
</tr>
<tr>
<td></td>
<td>Frame Structure</td>
<td>A-5</td>
</tr>
<tr>
<td></td>
<td>Flags</td>
<td>A-6</td>
</tr>
<tr>
<td></td>
<td>Address</td>
<td>A-6</td>
</tr>
<tr>
<td></td>
<td>Control Field</td>
<td>A-6</td>
</tr>
<tr>
<td>A.4</td>
<td>Path Control</td>
<td>A-9</td>
</tr>
<tr>
<td>A.5</td>
<td>Transmission Control</td>
<td>A-17</td>
</tr>
<tr>
<td>A.6</td>
<td>Data Flow Control</td>
<td>A-20</td>
</tr>
<tr>
<td>A.7</td>
<td>Function Management</td>
<td>A-20</td>
</tr>
<tr>
<td></td>
<td>SD (Sense Data)</td>
<td>A-20</td>
</tr>
<tr>
<td></td>
<td>RU (Request/Response Unit)</td>
<td>A-21</td>
</tr>
<tr>
<td>A.8</td>
<td>Application Layer</td>
<td>A-21</td>
</tr>
</tbody>
</table>

SDLC/SNA Programmer's Manual
# TABLE OF CONTENTS [continued]

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B</strong></td>
<td>DATA FORMATS</td>
<td>B-1</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>SDLC/SNA TERMINOLOGY</td>
<td>C-1</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>COMMAND SUMMARIES</td>
<td>D-1</td>
</tr>
<tr>
<td><strong>E</strong></td>
<td>CODING CONVENTIONS</td>
<td>E-1</td>
</tr>
<tr>
<td>E.1</td>
<td>Stack Comments</td>
<td>E-1</td>
</tr>
<tr>
<td>E.2</td>
<td>Stack Comment Abbreviations</td>
<td>E-2</td>
</tr>
<tr>
<td>E.3</td>
<td>Program Comments</td>
<td>E-2</td>
</tr>
<tr>
<td>E.4</td>
<td>Test Manager Constructs</td>
<td>E-3</td>
</tr>
<tr>
<td>E.5</td>
<td>Spacing and Indentation Guidelines</td>
<td>E-3</td>
</tr>
<tr>
<td>E.6</td>
<td>Colon Definitions</td>
<td>E-4</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td>COMMAND CROSS REFERENCE LIST</td>
<td>F-1</td>
</tr>
</tbody>
</table>

INDEX
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Sample Stack Comment</td>
<td>1-1</td>
</tr>
<tr>
<td>2-1</td>
<td>Monitor Configuration Menu</td>
<td>2-1</td>
</tr>
<tr>
<td>3-1</td>
<td>SNA Monitor Data Flow Diagram – Live Data</td>
<td>3-1</td>
</tr>
<tr>
<td>3-2</td>
<td>SNA Monitor Data Flow Diagram – Offline Processing</td>
<td>3-2</td>
</tr>
<tr>
<td>3-3</td>
<td>SNA Monitor Data Flow Diagram – Freeze Mode</td>
<td>3-4</td>
</tr>
<tr>
<td>4-1</td>
<td>SNA Monitor Data Flow Diagram – Decode</td>
<td>4-1</td>
</tr>
<tr>
<td>5-1</td>
<td>SNA Data Flow Diagram – Capture to RAM</td>
<td>5-1</td>
</tr>
<tr>
<td>6-1</td>
<td>SNA Data Flow Diagram – Recording to Disk</td>
<td>6-1</td>
</tr>
<tr>
<td>7-1</td>
<td>SNA Data Flow Diagram – Display and Print</td>
<td>7-1</td>
</tr>
<tr>
<td>7-2</td>
<td>Display Format Menu</td>
<td>7-2</td>
</tr>
<tr>
<td>8-1</td>
<td>Filter Setup Menu 1</td>
<td>8-1</td>
</tr>
<tr>
<td>8-2</td>
<td>Connection Diagram – Display Filters Activated</td>
<td>8-2</td>
</tr>
<tr>
<td>8-3</td>
<td>Filter Setup Menu 2</td>
<td>8-11</td>
</tr>
<tr>
<td>9-1</td>
<td>Emulation Configuration Menu</td>
<td>9-1</td>
</tr>
<tr>
<td>9-2</td>
<td>Station Setup Menu</td>
<td>9-8</td>
</tr>
<tr>
<td>10-1</td>
<td>SDLC Emulation Data Flow Diagram – Live Data</td>
<td>10-2</td>
</tr>
<tr>
<td>10-2</td>
<td>SDLC Emulation Data Flow Diagram – Offline Processing</td>
<td>10-3</td>
</tr>
<tr>
<td>10-3</td>
<td>SDLC Emulation Data Flow Diagram – Freeze Mode</td>
<td>10-4</td>
</tr>
<tr>
<td>11-1</td>
<td>SDLC Emulation Data Flow Diagram – Decode</td>
<td>11-1</td>
</tr>
<tr>
<td>13-1</td>
<td>Buffer Structure</td>
<td>13-18</td>
</tr>
<tr>
<td>A-1</td>
<td>Point-to-Point Configuration</td>
<td>A-2</td>
</tr>
<tr>
<td>A-2</td>
<td>Multipoint Configuration</td>
<td>A-3</td>
</tr>
<tr>
<td>A-3</td>
<td>SNA Layers</td>
<td>A-4</td>
</tr>
<tr>
<td>A-4</td>
<td>SDLC Frame Structure</td>
<td>A-5</td>
</tr>
<tr>
<td>A-5</td>
<td>Transmission Header for FID Type 0</td>
<td>A-10</td>
</tr>
<tr>
<td>A-6</td>
<td>Transmission Header for FID Type 1</td>
<td>A-11</td>
</tr>
<tr>
<td>A-7</td>
<td>Transmission Header for FID Type 2</td>
<td>A-12</td>
</tr>
<tr>
<td>A-8</td>
<td>Transmission Header for FID Type 3</td>
<td>A-13</td>
</tr>
<tr>
<td>A-9</td>
<td>Transmission Header for FID Type 4 (Bytes 0 to 7)</td>
<td>A-14</td>
</tr>
<tr>
<td>A-10</td>
<td>Transmission Header for FID Type 4 (Bytes 8-15)</td>
<td>A-15</td>
</tr>
<tr>
<td>A-11</td>
<td>Transmission Header for FID Type 4 (Bytes 16–25)</td>
<td>A-16</td>
</tr>
<tr>
<td>A-12</td>
<td>Transmission Header for FID Type F</td>
<td>A-16</td>
</tr>
<tr>
<td>A-13</td>
<td>BIU Format – Multiple PIU’s</td>
<td>A-17</td>
</tr>
<tr>
<td>A-14</td>
<td>Request Header</td>
<td>A-18</td>
</tr>
<tr>
<td>A-15</td>
<td>Response Header</td>
<td>A-19</td>
</tr>
<tr>
<td>A-16</td>
<td>Sense Data Format</td>
<td>A-20</td>
</tr>
<tr>
<td>B-1</td>
<td>NRZ and NRZI Data Encoding</td>
<td>B-1</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-1</td>
<td>Dual Window Commands</td>
<td>7-8</td>
</tr>
<tr>
<td>8-1</td>
<td>Frame Layer Filter Commands</td>
<td>8-10</td>
</tr>
<tr>
<td>8-2</td>
<td>Transmission Header Filter Commands</td>
<td>8-12</td>
</tr>
<tr>
<td>8-3</td>
<td>Request/Response Header Filter Commands</td>
<td>8-13</td>
</tr>
<tr>
<td>8-4</td>
<td>Sense Data Filter Commands</td>
<td>8-15</td>
</tr>
<tr>
<td>9-1</td>
<td>Effect of Clocking and Emulation Interface Selections on Bit Rate</td>
<td>9-4</td>
</tr>
<tr>
<td>12-1</td>
<td>Primary Emulation States</td>
<td>12-2</td>
</tr>
<tr>
<td>12-2</td>
<td>Secondary Emulation States</td>
<td>12-2</td>
</tr>
<tr>
<td>13-1</td>
<td>Frame (BLU) Identifiers</td>
<td>13-4</td>
</tr>
<tr>
<td>13-2</td>
<td>Transmission Header Identifiers</td>
<td>13-7</td>
</tr>
<tr>
<td>13-3</td>
<td>Request/Response Unit Category Identifiers</td>
<td>13-8</td>
</tr>
<tr>
<td>13-4</td>
<td>FMD Request/Response Unit Identifiers</td>
<td>13-9</td>
</tr>
<tr>
<td>13-5</td>
<td>DFC Request/Response Unit Identifiers</td>
<td>13-12</td>
</tr>
<tr>
<td>13-6</td>
<td>SC Request/Response Unit Identifiers</td>
<td>13-12</td>
</tr>
<tr>
<td>13-7</td>
<td>NC Request/Response Unit Identifiers</td>
<td>13-13</td>
</tr>
<tr>
<td>13-8</td>
<td>V.28/RS-232C Interface Lead Transitions</td>
<td>13-17</td>
</tr>
<tr>
<td>13-9</td>
<td>V.35 Interface Lead Transitions</td>
<td>13-17</td>
</tr>
<tr>
<td>13-10</td>
<td>V.36/RS-449 Interface Lead Transitions</td>
<td>13-17</td>
</tr>
<tr>
<td>A-1</td>
<td>SDLC Commands and Responses – Modulo 8</td>
<td>A-8</td>
</tr>
<tr>
<td>A-2</td>
<td>SDLC Commands and Responses – Modulo 128</td>
<td>A-9</td>
</tr>
<tr>
<td>B-1</td>
<td>Clocking Modes</td>
<td>B-1</td>
</tr>
<tr>
<td>D-1</td>
<td>Physical Events</td>
<td>D-1</td>
</tr>
<tr>
<td>D-2</td>
<td>Setting Leads</td>
<td>D-1</td>
</tr>
<tr>
<td>D-3</td>
<td>Frame Identifiers</td>
<td>D-2</td>
</tr>
<tr>
<td>D-4</td>
<td>Frame Events</td>
<td>D-2</td>
</tr>
<tr>
<td>D-5</td>
<td>SNA Header Identifiers</td>
<td>D-3</td>
</tr>
<tr>
<td>D-6</td>
<td>SNA Header Events</td>
<td>D-4</td>
</tr>
<tr>
<td>D-7</td>
<td>Sending Frames and BTU's</td>
<td>D-4</td>
</tr>
<tr>
<td>D-8</td>
<td>Creating Buffers</td>
<td>D-5</td>
</tr>
<tr>
<td>D-9</td>
<td>Starting and Examining Timers</td>
<td>D-5</td>
</tr>
<tr>
<td>D-10</td>
<td>Timer Events</td>
<td>D-5</td>
</tr>
<tr>
<td>D-11</td>
<td>Creating User Output</td>
<td>D-6</td>
</tr>
<tr>
<td>D-12</td>
<td>Program Control Events</td>
<td>D-7</td>
</tr>
<tr>
<td>E-1</td>
<td>ITL Symbols</td>
<td>E-2</td>
</tr>
</tbody>
</table>
The SNA Monitor is implemented in accordance with IBM's implementation of SNA for single link or multi-link procedures. It is not a state driven—monitor, i.e. it does not have the knowledge of expected events. Rather, the monitor decodes and reports information on received frames and lead changes. Filters, RAM capture, and disk recording are also available.

The SDLC Emulation is implemented as a state—driven protocol emulation together with an integral protocol monitor. The emulation has been set up to run as:

- an automatic simulation which operates in accordance with IBM’s SDLC procedure for modulo 8;
- up to 32 simultaneous links;
- a semi—automatic tester. The test manager is used to build and execute test scenarios for generation of errors and to test responses; and
- a manual tester. The test is controlled from the user’s keyboard.

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.

![Figure 1-1 Sample Stack Comment](image)

See Appendix E for further explanation of stack parameters.

Sample test scripts are supplied in Section 14. These test scripts are also supplied on disk with the application program.

The SDLC/SNA applications can be controlled remotely from a terminal. All commands described in this manual can be entered from the remote terminal’s keyboard followed by a <cr> (RETURN). The application processes the remote command and returns the ‘ROK’ prompt to the remote terminal. The remote terminal must be connected to the modem port on the back of the tester. To configure the application for remote control, refer to the Programmer’s Reference Manual.
This section describes the commands associated with each item on the Monitor Configuration Menu.

![Monitor Configuration Menu]

- **Interface Type**
  - RS232C/V.28
  - Interface Leads: DISABLED
  - Bit Rate: UNKNOWN
  - Clocking: NRZ WITH CLOCK
  - Frame Sequence Number Modulo: MOD 8

**Figure 2-1 Monitor Configuration Menu**

**WAKEUP_CPU** (---)
Initializes the SNA protocol for the monitor and configures the physical interface.

**NOTE**
Use WAKEUP_CPU once after all physical changes are made.

**Interface Type**

- **IF=V28** (---)
  Selects the V.28/RS-232C connector (default) 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 and electrically isolates the other connectors on the port.

  - V.35 function key
IF=V36 (---)
Selects the V.36/RS-449 connector for protocol testing 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.

→ Interface Leads
Individual or all interface leads can be enabled or disabled (default). Leads must be enabled for test manager detection.

ENABLE_LEAD (lead identifier ---)
Enables the specified lead. Refer to the Programmer’s Reference Manual for a list of supported leads for each interface type.

Example:
Enable the request to send lead.
IRS ENABLE_LEAD

DISABLE_LEAD (lead identifier ---)
Disables (default) the specified lead. Refer to the Programmer’s Reference Manual for a list of supported leads for each interface type.

Example:
Disable the clear to send lead.
ICS DISABLE_LEAD

ALL_LEADS (---lead identifier)
Enables/disables all leads supported on the currently selected WAN interface. ALL_LEADS must be used with ENABLE_LEAD or DISABLE_LEAD.

Example 1:
Enable all leads on the current interface.
ALL_LEADS ENABLE_LEAD

ENABLED function key

Example 2:
Disable all leads on the current interface.
ALL_LEADS DISABLE_LEAD

DISABLED function key
\textbf{→ Bit Rate}

When NRZI clocking is selected, the interface speed can be selected from preset values on the Interface Port Speed Menu or set to a user-defined speed. When any other clocking mode is selected, the speed is measured, in bits per second, directly from the physical line.

The following commands should only be used when NRZI clocking is selected.

\begin{verbatim}
SPEED=50  SPEED=300  SPEED=4800  SPEED=38400
SPEED=75  SPEED=1200 SPEED=7200  SPEED=48000
SPEED=110 SPEED=1800 SPEED=9600 (default) SPEED=56000
SPEED=134.5 SPEED=2000 SPEED=14400 SPEED=64000
SPEED=150 SPEED=2400 SPEED=16000 SPEED=72000
SPEED=200 SPEED=3600 SPEED=19200 SPEED=128000
\end{verbatim}

\textbf{INTERFACE–SPEED ( -- address )}

Contains the current bit rate (default value is 9600) and is used by the monitor to calculate throughput measurements.

\textbf{CK\_LEAD\_ENABLE ( -- )}

Enables the detection of control leads for the currently selected WAN interface.

\textbf{→ Clocking}

\textbf{IF=NRZ ( -- )}

Selects standard non-return to zero line encoding (default) with DCE provided clocks.

\textbf{NRZ WITH CLOCK function key}

\textbf{IF=NRZ\_EXT ( -- )}

Selects a DTE provided transmit clock on pint 24 of an RS-232C connector.

\textbf{EXTERNAL TX CLOCK function key}

\textbf{IF=NRZI ( -- )}

Selects the non-return to zero inverted method of encoding with timing information extracted from the data signal.

\textbf{NRZI function key}

\textbf{IF=NRZI+CLK ( -- )}

Selects the non-return to zero inverted method of encoding with timing information extracted from the provided clock signal.

\textbf{NRZI WITH CLOCK function key}

\textbf{NOTE}

\textit{If any of these commands are used when the application is running on a B–Channel, the clocking is forced to IF=NRZ (standard non–return to zero encoding).}
Frame Sequence Number Modulo

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

MOD 8 function key

L2=MOD128 (---)
Enables modulo 128 method of decoding.

MOD 128 function key

NOTE
The program is automatically placed into modulo 8 when an SNRM is received, and modulo 128 when an SNRME is received.
The SNA 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 data displayed, captured, or recorded data.

### 3.1 Live Data

The monitor application receives events from the interface or from the internal timer and processes them as shown in Figure 3–1.

![Figure 3-1 SNA Monitor Data Flow Diagram – Live Data](image)

By default, the SNA Monitor captures data in the capture RAM buffer and displays it on the screen in a short format report.
MONITOR ( -- )
Selects the live data mode of operation. All incoming events are decoded and displayed in real-time.

### 3.2 Playback

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

![Figure 3-2 SNA Monitor Data Flow Diagram - Offline Processing](image)

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 to 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 open for disk recording or disk playback.

Example:
Obtain playback data from disk.

FROM_DISK ( Identify a Disk file as data source )
HALT ( Place the monitor in playback mode )
" SNADAT" =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.

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

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

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

TOP ( -- )
Positions the screen at the beginning of the playback source.
3.3 Simultaneous Live Data and Playback

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

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.
The SNA Monitor supports both modulo 8 and modulo 128 decoding and display.

Data or lead changes from the interface, capture RAM, or disk file are decoded by protocol layer. Each decoding layer stores information in a pool of variables for later use by either a test program or other parts of the monitor application.

4.1 Communication Variables

The following variables are set during the decode process and contain protocol specific information as defined by IBM.

\[ \text{NOTE} \]
These variables can be read using the @ (fetch) operation.
Layer 1

The layer 1 decode operation saves information concerning frame length, timestamps, port identifier, and block sequence number. For lead transitions, information is saved concerning the changed lead(s); for timers, the number of the expired timer.

PORT-ID (-- address )
Contains a 2 byte value identifying received direction for data. The lower byte indicates the TO_DCE or TO_DTE receive stream. The upper byte indicates the application processor that received the frame.

Example:
Determine the direction of the received stream.
PORT-ID @
0xFF AND ( The AND operation eliminates the upper byte )

This operation leaves the received stream direction on the stack. It is 0 for a trace statement or equal to one of the following pre-defined constants: TO_DTE_RX for data to the terminal or TO_DCE_RX for data to the network. For further explanation of port identification, consult the Programmer's Reference Manual.

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

Example:
Obtain the start of frame timestamp including year, month, day, hour, minute, second, and millisecond.
START-TIME GET_TSTAMP_MILLI

NOTE
The @ (fetch) operation is not performed. Seven values are left on the stack as described in the Programmer's Reference Manual.

END-TIME (--address )
Contains the 48 bit end of frame timestamp for data. Use with the GET_TSTAMP_MILLI or GET_TSTAMP_MICRO command. See the START-TIME example.

BLOCK-COUNT (-- address )
Contains the sequential block sequence number for live data. Every received frame is assigned a unique sequence number. Each side, DTE or DCE, maintains a separate set of sequence numbers. Initially contains a value of zero and is incremented by one each time a new block is received.

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

Example:
REC-LENGTH @ provides the total number characters in the current frame (BLU).
**REC-POINTER** ( -- address )
Contains the pointer to the frame address field (first byte) in the received frame. Since this variable contains the address of the first byte, a double fetch operation is necessary to obtain frame contents.

Example:
Obtain the second byte of the received frame (the control field).

```
REC-POINTER C@  
```

**NOTE**
The `@` command gets the address of the first byte in the received frame. This first value is then incremented by one and one byte is fetched from the resulting address.

**STATUS_ERR?** ( -- flag )
Returns true if an error is detected in the currently processed frame. Use the following commands to detect a particular error.

<table>
<thead>
<tr>
<th>Command</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVERRUN_ERR?</td>
<td>Receiver overrun</td>
</tr>
<tr>
<td>CRC_ERR?</td>
<td>CRC error</td>
</tr>
<tr>
<td>ABORT_ERR?</td>
<td>Abort error</td>
</tr>
<tr>
<td>LONG_FRM_ERR?</td>
<td>Frame is longer than XXXX bytes</td>
</tr>
<tr>
<td>SHORT_FRM_ERR?</td>
<td>Frame is shorter than XX bytes (including 2 CRC bytes)</td>
</tr>
<tr>
<td>CTRL_BYTE_ERR?</td>
<td>Invalid control byte</td>
</tr>
<tr>
<td>ADDR_BYTE_ERR?</td>
<td>Invalid address byte</td>
</tr>
<tr>
<td>LENGTH_ERR?</td>
<td>Invalid frame length</td>
</tr>
</tbody>
</table>

**LEAD-NUMBER** ( -- address )
Contains the received lead identifier used in the test manager.

---

**Frame (SDLC) Layer**
The frame (SDLC) layer decode operation saves information concerning a frame’s address, control byte and with I frames, sets up the pointer and length for the transmission header decode.

**FRAME-ADDR** ( -- address )
Contains the PU address (first byte) of the received frame. Valid values are 0 through 255.

**M-CONTROL** ( -- address )
Contains the received frame control field. In frame modulo 8, the control field is the second byte of the received frame. In frame modulo 128, the control field is the second byte for unnumbered frames, and second and third bytes for I frames and supervisory frames. See Tables A-1 and A-2 for possible values.

**FRAME-MODULO** ( -- address )
Contains 0 (default) for modulo 8, and 1 for modulo 128. FRAME-MODULO is set to 0 when an SNRM is received, and 1 when an SNRME is received.
M-PF ( -- address )
Contains the poll/final bit for the received frame (0 or 1). In frame modulo 8, the poll/final bit is bit 3 of the control field. In frame modulo 128, the poll/final bit is bit 3 of the control field for unnumbered frames and bit 15 of the control field in information and supervisory frames.

M-NR ( -- address )
Contains the N(R) (receive sequence count) value of the received frame. Valid values are 0 through 7 for frame modulo 8, and 0 through 127 for frame modulo 128. In frame modulo 8, the N(R) values are bits 0 to 2 of the control field for information and supervisory frames. In frame modulo 128, the N(R) values are bits 8 to 14 of the control field for information and supervisory frames.

M-NS ( -- address )
Contains the N(S) (send sequence count) value of the received frame. Valid values are 0 through 7 for frame modulo 8, and 0 through 127 for frame modulo 128. In frame modulo 8, the N(S) values are bits 4 to 6 of the control field for information frames. In frame modulo 128, the N(S) values are bits 0 to 6 of the control field for information frames.

FID-LENGTH ( -- address )
Contains the length of the received transmission header and any following bytes. This length does not include the frame layer bytes or the FCS bytes. If the received frame is not an I frame, the length is 0.

FID-POINTER ( -- address )
Contains the pointer to byte 0 in the received transmission header.

TH (Transmission Header)
SNA defines six types of TH's (transmission headers): FID0, FID1, FID2, FID3, FID4, and FIDF. The format of each type is determined by the FID (format identifier) which is located in bits 0 to 3 of byte 0 of the TH.

\[ \text{NOTE} \]
Refer to Appendix A.4 for the format of these headers.
Common FID Variables

FID-ID ( -- address )
Contains the format identifier field of the transmission header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*FID0</td>
<td>FID Type 0</td>
</tr>
<tr>
<td>R*FID1</td>
<td>FID Type 1</td>
</tr>
<tr>
<td>R*FID2</td>
<td>FID Type 2</td>
</tr>
<tr>
<td>R*FID3</td>
<td>FID Type 3</td>
</tr>
<tr>
<td>R*FID4</td>
<td>FID Type 4</td>
</tr>
<tr>
<td>R*FIDF</td>
<td>FID Type F</td>
</tr>
<tr>
<td>R*INVFID</td>
<td>Invalid FID</td>
</tr>
</tbody>
</table>

MAPPING-FIELD ( -- address )
Contains the mapping field located in bits 4 and 5 of byte 0 of FID0, FID1, FID2, and FID3, and bits 4 and 5 of byte 16 of FID4. This field indicates if the message is segmented and which part of the segment it is.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*WHOLE-BIU</td>
<td>Whole message</td>
</tr>
<tr>
<td>R*FIRST-SEGMENT</td>
<td>First segment of message</td>
</tr>
<tr>
<td>R*LAST-SEGMENT</td>
<td>Last segment of message</td>
</tr>
<tr>
<td>R*MIDDLE-SEGMENT</td>
<td>Middle or between segment of message</td>
</tr>
</tbody>
</table>

EXP-FLOW-IND ( -- address )
Contains the EFI (expedited flow indicator) located in bit 7 of byte 0 of FID0, FID1, FID2, and FID3, and bit 7 of byte 16 of FID4. The EFI informs the LU transmission control layer that the message is to be handled in an expedited fashion.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal</td>
</tr>
<tr>
<td>1</td>
<td>Expedited</td>
</tr>
</tbody>
</table>

DEST-ADD-FIELD ( -- address )
Contains the DAF (destination address field). This field, located in bytes 2 and 3 of FID0 and FID1, and byte 2 of FID2, specifies which mode the message is directed. Valid values are 0 through 65535.

ORIG-ADD-FIELD ( -- address )
Contains the OAF (origination address field). This field, located in bytes 4 and 5 of FID0 and FID1, and byte 3 of FID2 specifies the mode transmitting the message. Valid values are 0 through 65535.

SEQ-NUM-FIELD ( -- address )
Contains the SNF (sequence number field) located in bytes 6 and 7 of FID0 and FID1, bytes 4 and 5 of FID2, and bytes 22 and 23 of FID4. This field is generated by the LU (logical unit) and is used for detection of lost messages. Valid values are 0 through 65535.
DATA-COUNT-FIELD (-- address)
Contains the DCF (data count field) located in bytes 8 and 9 of FID0 and FID1, and bytes 24 and 25 of FID4 and FIDF. The field specifies how many bytes of data comprise the RH (request/response header) and RU (request/response unit).

FID3 Variables

LU–SSCP–IND (-- address)
Contains the LU/SSCP indicator from the LSID (local system ID) located in bit 0 of byte 1 of FID3. The LU/SSCP indicates whether the source of the message was an SSCP (system services control point) or an LU (logical unit).

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SSCP is the message source</td>
</tr>
<tr>
<td>80</td>
<td>LU is the message source</td>
</tr>
</tbody>
</table>

LU–PU–IND (-- address)
Contains the LU/PU indicator from the LSID (local system ID) located in bit 1 of byte 1 of FID3. The LU/PU indicates whether the destination of the message is a PU (physical unit) or an LU (logical unit).

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Destination is a PU</td>
</tr>
<tr>
<td>40</td>
<td>LU is the message source</td>
</tr>
</tbody>
</table>

LOCAL–ADDRESS (-- address)
Contains the local address field from the LSID (local system ID) located in bits 2 to 7 of byte 1 of FID3. This field indicates which LU (logical unit) the message is destined. Valid values are 0 through 63.

FID4 Variables

TG–Sweep (-- address)
Contains the transmission group sweep indicator located in bit 4 of byte 0 of FID4. Specifies whether PIU's (path information unit) can pass PIU's.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PIU's may pass PIU's</td>
</tr>
<tr>
<td>8</td>
<td>PIU's may not pass PIU's</td>
</tr>
</tbody>
</table>
ER–VR–SUPP–IND ( -- address )
Contains the ER (explicit route) and VR (virtual route) support indicator located in bit 5 of byte 0 of FID4.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Each node supports ER and VR</td>
</tr>
<tr>
<td>4</td>
<td>Each node does not support ER and VR</td>
</tr>
</tbody>
</table>

VR–PAC–CNT–IND ( -- address )
Contains the VR (virtual route) pacing count located in bit 6 of byte 0 of FID4. Pacing is the method by which a receiving component controls the rate of transmission of a sending unit to prevent overrun or congestion.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>VR pacing count not zero</td>
</tr>
<tr>
<td>2</td>
<td>VR pacing count is zero</td>
</tr>
</tbody>
</table>

NTWK–PRIORITY ( -- address )
Contains the network priority indicator located in bit 7 of byte 0 of FID4. Indicates whether the flow is less than network priority.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Flow is less than network priority</td>
</tr>
<tr>
<td>1</td>
<td>Flow is network priority</td>
</tr>
</tbody>
</table>

IERN ( -- address )
Contains the value of the IERN (initial explicit route number) located in bits 0 to 3 of byte 2 of FID4. Valid values are 0 through 15.

ERN ( -- address )
Contains the value of the ERN (explicit route number) located in bits 4 to 7 of byte 2 of FID4. Valid values are 0 through 15.

VRN ( -- address )
Contains the VRN (virtual route number) located in bits 0 to 3 of byte 3 of FID4. Valid values are 0 through 15.

TPF ( -- address )
Contains the TPF (transmission priority field) located in bits 6 and 7 of byte 3 of FID4.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Low transmission priority</td>
</tr>
<tr>
<td>1</td>
<td>Medium transmission priority</td>
</tr>
<tr>
<td>2</td>
<td>High transmission priority</td>
</tr>
</tbody>
</table>
VR-CWI (-- address)
Contains the VR CWI (virtual route change window indicator) located in bit 0 of byte 4 of FID4.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Increment window size</td>
</tr>
<tr>
<td>80</td>
<td>Decrement window size</td>
</tr>
</tbody>
</table>

TG-NON-FIFO-IND (-- address)
Contains the TG FIFO (transmission group FIFO indicator) located in bit 1 of byte 1 of FID4.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TG FIFO required</td>
</tr>
<tr>
<td>40</td>
<td>TG FIFO not required</td>
</tr>
</tbody>
</table>

VR-SQTI (-- address)
Contains the VR SQTI (virtual route sequence and type indicator) located in bits 2 and 3 of byte 4 of FID4.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Non-sequenced, non-supervisor</td>
</tr>
<tr>
<td>10</td>
<td>Non-sequenced, supervisor</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
</tr>
<tr>
<td>30</td>
<td>Singly sequenced</td>
</tr>
</tbody>
</table>

TG-SNF (-- address)
Contains the TGSNF (transmission group sequence number) located in bits 4 to 7 of byte 4 and all of byte 5 of FID4. Valid values are 0 through 4095.

VR-PRQ (-- address)
Contains the VR PRQ (virtual route pacing request) located in bit 0 of byte 6 of FID4.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No VR pacing response requested</td>
</tr>
<tr>
<td>80</td>
<td>VR pacing response requested</td>
</tr>
</tbody>
</table>

VR-PRS (-- address)
Contains the VR PRS (virtual route pacing response) located in bit 1 of byte 6 of FID4.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No VR pacing response</td>
</tr>
<tr>
<td>40</td>
<td>VR pacing response sent</td>
</tr>
</tbody>
</table>
VR–CWRI ( -- address )
Contains the VR CWRI (virtual route change window reply indicator) located in bit 2 of byte 6 of FID4.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Increment window size by 1</td>
</tr>
<tr>
<td>20</td>
<td>Decrement window size by 1</td>
</tr>
</tbody>
</table>

VR_RWI ( -- address )
Contains the VR RWI (virtual route reset window reply indicator) located in bit 3 of byte 6 of FID4.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not reset window size</td>
</tr>
<tr>
<td>10</td>
<td>Reset window size to minimum</td>
</tr>
</tbody>
</table>

VR–SNF–SEND ( -- address )
Contains the virtual route send sequence number located in bits 4 to 7 of byte 6 and all of byte 7 of FID4. Valid values are 0 through 4095.

DSAF ( -- address )
Contains the DSAF (destination sub–area address) located in bytes 8 to 11 of FID4.

OSAF ( -- address )
Contains the OSAF (origin sub–area address) located in bytes 12 to 15 of FID4.

SNAI ( -- address )
Contains the SNA indicator located in bit 4 of byte 16 of FID4.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not SNA</td>
</tr>
<tr>
<td>10</td>
<td>SNA</td>
</tr>
</tbody>
</table>

DEF ( -- address )
Contains the DEF (destination element field) located in bytes 18 and 19 of FID4.

OEF ( -- address )
Contains the OEF (origin element field) located in bytes 20 and 21 of FID4.

FIDF Variables
COMMAND–FORMAT ( -- address )
Contains the command format field located in byte 2 of FIDF. Valid values are 0 through 255.

COMMAND–TYPE ( -- address )
Contains the command type located in byte 3 of FIDF. Valid values are 0 through 255.
COMMAND-SEQUENCE-NUMBER ( -- address )
Contains the command sequence number located in bytes 4 and 5 of FIDF.

RH (Request/Response Header)

RH-POINTER ( -- address )
Contains the pointer to the first byte of the request/response header.

RH-LENGTH ( -- address )
Contains the length of the request/response header (3 bytes), plus the length of sense data if present (4 bytes), plus the length of the request/response unit and user data.

RH-REQ-RES ( -- address )
Contains the request/response indicator located in bit 8 of byte 0 of the request/response header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*RES-HDR</td>
<td>Response</td>
</tr>
<tr>
<td>R*REQ-HDR</td>
<td>Request</td>
</tr>
</tbody>
</table>

RH-ID ( -- address )
Contains the request/response unit category located in bits 6 and 7 of byte 0 of the request/response header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*FMD</td>
<td>FMD (function management data)</td>
</tr>
<tr>
<td>R*NC</td>
<td>NC (network control)</td>
</tr>
<tr>
<td>R*DFC</td>
<td>DFC (data flow control)</td>
</tr>
<tr>
<td>R*SC</td>
<td>SC (session control)</td>
</tr>
<tr>
<td>R*INV RH</td>
<td>Invalid RH</td>
</tr>
</tbody>
</table>
FORMAT-IND ( -- address )
Contains the FI (format indicator) located in bit 4 of byte 0 of the request/response header. This indicator defines field formatting for either a LU-LU session or SSCP (network services) session.

<table>
<thead>
<tr>
<th>Value</th>
<th>LU-LU Session</th>
<th>SSCP Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No FM header follows</td>
<td>Character coded RU (request/response unit)</td>
</tr>
<tr>
<td>8</td>
<td>FM header follows</td>
<td>Field formatted RU</td>
</tr>
</tbody>
</table>

SENSE-DATA-IND ( --address )
Contains the SDI (sense data included indicator) located in bit 5 of byte 0 of the request/response header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sense data not included</td>
</tr>
<tr>
<td>4</td>
<td>Sense data included (i.e. an error has occurred)</td>
</tr>
</tbody>
</table>

CHAIN-IND ( -- address )
Contains the BCI and ECI (begin and end chain indicators) located in bits 6 and 7 of byte 0 of the request header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Middle RU in the chain</td>
</tr>
<tr>
<td>1</td>
<td>Last RU in the chain</td>
</tr>
<tr>
<td>2</td>
<td>First RU in the chain</td>
</tr>
<tr>
<td>3</td>
<td>Only RU in the chain</td>
</tr>
</tbody>
</table>

RESPONSE-TYPE ( -- address )
Contains the DR1I and DR2I (definite response indicators 1 and 2), and the ERI (exception response indicator) for a request header. For a response header, it contains the DR1I and DR2I and the RTI (response type indicator). These indicators are located in bits 0, 2, and 3 respectively of byte 1 of the request/response header.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Request Header</th>
<th>Response Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No response requested</td>
<td>Reserved</td>
</tr>
<tr>
<td>20</td>
<td>Definite response requested by DR2I</td>
<td>Positive response to DR2I</td>
</tr>
<tr>
<td>30</td>
<td>Exception response requested by DR2I</td>
<td>Negative response to DR2I</td>
</tr>
<tr>
<td>80</td>
<td>Definite response requested by DR1I</td>
<td>Positive response to DR1I</td>
</tr>
<tr>
<td>90</td>
<td>Exception response requested by DR1I</td>
<td>Negative response to DR1I</td>
</tr>
<tr>
<td>A0</td>
<td>Definite response requested by DR1I and DR2I</td>
<td>Positive response to DR1I and DR2I</td>
</tr>
<tr>
<td>B0</td>
<td>Exception response requested by DR1I and DR2I</td>
<td>Negative response to DR1I and DR2I</td>
</tr>
</tbody>
</table>
QUEUED-RESP-IND ( -- address )
Contains the QRI (queued response indicator) located in bit 6 of byte 1 of the request/response header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Response bypasses transmission control queues</td>
</tr>
<tr>
<td>2</td>
<td>Enqueues response in transmission control queues</td>
</tr>
</tbody>
</table>

PACING-IND ( -- address )
Contains the PI (pacing indicator) located in bit 7 of byte 1 of the request/response header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not a pacing request</td>
</tr>
<tr>
<td>1</td>
<td>Pacing request or response to pacing request</td>
</tr>
</tbody>
</table>

BRACKET-IND ( -- address )
Contains the BBI (begin bracket indicator) and EBI (end bracket indicator) fields located in bits 0 and 1 respectively of byte 2 of the request header.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Between or within brackets</td>
</tr>
<tr>
<td>40</td>
<td>Notification to other half session that the speaker is finished and that the bracket protocol is complete</td>
</tr>
<tr>
<td>80</td>
<td>Request by a half session to become speaker of the session</td>
</tr>
<tr>
<td>C0</td>
<td>Only chain in the bracket</td>
</tr>
</tbody>
</table>

CHANGE-DIR-IND ( -- address )
Contains the CDI (change direction indicator) located in bit 2 of byte 2 of the request header. This indicator is used to avoid contention within HDX–FF (half duplex flip–flop) sessions.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No change in direction requested</td>
</tr>
<tr>
<td>20</td>
<td>Indicates the sending LU is finished and requests that the receiving LU now become the sending LU (i.e. change direction)</td>
</tr>
</tbody>
</table>

CODE-SEL-IND ( -- address )
Contains the CSI (code selection indicator) located in bit 4 of byte 2 of the request header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Code 0</td>
</tr>
<tr>
<td>8</td>
<td>Code 1</td>
</tr>
</tbody>
</table>
ENCIPH-DATA-IND (-- address)
Contains the EDI (enciphered data indicator) located in bit 5 of byte 2 of the request header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RU is not enciphered</td>
</tr>
<tr>
<td>4</td>
<td>RU is enciphered</td>
</tr>
</tbody>
</table>

PADDED-DATA-IND (-- address)
Contains the PDI (padded data indicator) located in bit 6 of byte 2 of the request header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RU is not padded</td>
</tr>
<tr>
<td>2</td>
<td>RU was padded to a length which is an integral multiple of 8 bytes before enciphering</td>
</tr>
</tbody>
</table>

SD (Sense Data)
The decode operation for sense data is called only when the value in the sense data included field of the RH header contains a 1. In this case, the SENSE-DATA-IND variable contains the value of 4.

SD-POINTER (-- address)
Contains the pointer to byte 0 of the sense data.

SD-LENGTH (-- address)
Contains the remaining length of the frame starting at the first byte of the sense data field.
This includes 4 bytes of sense data, the request/response unit field, and any user data.

Example:
Obtain the modifier in byte 1 of the sense data field.

SD-LENGTH @ 3 > (Check to see if sense data field is complete)
IF
    SD-POINTER @ 1+ C@ (Get the modifier byte)
ENDIF
**SD-ID ( -- address )**
Contains the category of sense data located in byte 0 of the sense data field.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*USER-SD</td>
<td>User sense data only</td>
</tr>
<tr>
<td>R*REQREJ-SD</td>
<td>Request reject</td>
</tr>
<tr>
<td>R*REQERR-SD</td>
<td>Request error</td>
</tr>
<tr>
<td>R*STERR-SD</td>
<td>State error</td>
</tr>
<tr>
<td>R*RHERR-SD</td>
<td>Request header usage error</td>
</tr>
<tr>
<td>R*PERR-SD</td>
<td>Path error</td>
</tr>
<tr>
<td>R*INVSD</td>
<td>Invalid sense data</td>
</tr>
</tbody>
</table>

⚠️ **NOTE**
A value of 0 in this variable indicates that sense data is not present.

Example:
Obtain the sense-code specific information in bytes 2 and 3 of the sense data field.

```
SD-LENGTH @ 3 >  ( Check to see if sense data field is complete )
IF
  SD-POINTER @ 2+ w@  ( Obtain 2 bytes )
ENDIF
```

---

**RU (Request/Response Unit)**

**RU-POINTER ( -- address )**
Contains the pointer to byte 0 of the request/response unit.

⚠️ **NOTE**
Request/response units contained within an FMD transmission header have a 3 byte network services header.

Example:
Obtain the network services header.

```
RH-ID @ R*FMD =  ( Is the TH an FMD? )
IF
  RU-POINTER @ @  ( Obtain the NS - 32 bits header )
  8 >>#  ( Shift 8 bits to the right to obtain the three byte header )
ENDIF
```

⚠️ **WARNING**
Request/response units within an FMD transmission header that are character coded, or LU to LU requests, are implementation dependent. The code in the above example should not be used in these cases.
Example:
Obtain the 1 byte request code of the request/response units contained within a DFC, SC, or NC transmission header.

RH-ID @ R*DFC =
IF
   RU-POINTER @ C@
ENDIF

RU-LENGTH ( -- address )
Contains the remaining length of the frame starting at the first byte of the request/response unit. This length includes the length of the request/response unit field and the length of any user data.
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 3.2. Furthermore, data stored in capture RAM can be transferred to disk.

![SNA Data Flow Diagram - Capture to RAM](image)

**Figure 5-1** SNA Data Flow Diagram – Capture to RAM

### 5.1 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 old data (default) after the capture buffer is full (endless loop recording).

⚠️ WARNING  
CAPT_FULL and CAPT_WRAP erase all data in capture RAM.

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

5.2 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.

QUIT_TRA ( -- )  
Abruptly terminates the transfer of data from capture RAM to disk.
**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  
HALT  
"DATA1" =TITLE  
RECORD  
CTOD_ON  
TRA_ALL  
TRANSFER  
DISK_OFF  

( Designate Capture RAM as data source )
( Enter playback mode )
( Assign filename DATA1 )
( Open data recording )
( Enable Capture Transfer to disk )
( Transfer all data )
( Transfer data from Capture to disk )
( 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 ( Designate Capture RAM as data source )
HALT ( Enter playback mode )
PRINT_ON ( Enable printing )
TRA_ALL ( Transfer all )
TRANSFER ( 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 3.2. Data stored in capture RAM can be transferred to disk as described in Section 5.2.

**Figure 6-1** SNA Data Flow Diagram – Recording to Disk

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

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

**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 disk), 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 : Universal Mon Vl.3-1.3 Rev 0 WAN RS232-C PT500 - 24 SN# 03-1
Recording End : Universal Sim Vl.3-1.3 Rev 0 WAN RS232-C PT500 - 24 SN# 03-1

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 data 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 SNA Monitor and SDLC Emulation applications can display data from the line (live data), from capture RAM, or from a disk recording the following four basic display formats:

- Hexadecimal
- Character
- Short
- Complete
- Trace Statements

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

![Data Flow Diagram](image)

**Figure 7-1 SNA Data Flow Diagram – Display and Print**

**NOTE**
*Data can only be printed in playback mode.*
Display Format Menu

<table>
<thead>
<tr>
<th>Display Format COMPLETE</th>
<th>Dual Window</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>OFF</td>
<td>Trace Display Format</td>
</tr>
<tr>
<td>Character Set</td>
<td>EBCDIC</td>
<td>Throughput Graph</td>
</tr>
<tr>
<td>Frame Layer</td>
<td>TEXT</td>
<td>Short Interval (sec)</td>
</tr>
<tr>
<td>Transmission Header</td>
<td>TEXT</td>
<td>Long Interval (sec)</td>
</tr>
<tr>
<td>Request/Resp Header</td>
<td>TEXT</td>
<td></td>
</tr>
<tr>
<td>Sense Data</td>
<td>TEXT</td>
<td></td>
</tr>
<tr>
<td>Request/Resp Unit</td>
<td>TEXT</td>
<td></td>
</tr>
<tr>
<td>Data Field</td>
<td>CHARACTER</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7-2 Display Format Menu

→ Display Format
The default display is short format. Frame Layer, Transmission Header, Request/Response Header, Sense Data, Request/Response Unit, and Data Field can only be modified when Display Format is set to COMPLETE.

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

عش OFF function key (not highlighted)

REP_OFF (---)
Turns off data display.

عش OFF function key (highlighted)

REP_COMP (---)
Displays data in a comprehensive report. Each protocol layer has its own display generator and thus can be turned on, off, or selected as text, hex, or character display.

عش COMPLETE function key

REP_SHORT (---)
Displays data in a condensed report (default). This includes the source identifier, the PU address, the frame type, the transmission header, request/response header, and request/response unit. This format is useful for higher speed monitoring as more frames per screen are displayed and processing is kept to a minimum.

عش SHORT function key
DISPLAY FORMAT

September 1990

REP_HEX (--) 
Displays timestamps or the block sequence numbers, and the port identifier in text. Frame contents are displayed in hex.

epad HEX function key

REP_CHAR (--) 
Displays timestamps or the block sequence numbers, and the port identifier in text. Frame contents are displayed in the currently selected character set.

epad CHARACTER function key

REP_NONE (--) 
Displays only trace statements.

epad Format topic

TRACE function key

→ Timestamp
Timestamp reporting is available when display format is not in short mode.

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

epad OFF function key

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

epad MM:SS:ssss function key

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

epad DD HH:MM:SS function key
→ **Character Set**
Selects the character set for data display.

**R=ASCII (--)**
Sets the character set for data display to ASCII.

ASCII function key

**R=EBCDIC (--)**
Sets the character set for data display to EBCDIC (default).

EBCDIC function key

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

HEX function key

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

JIS8 function key

→ **Frame Layer**
**FRM_ON (--)**
Displays frame (SDLC) layer data in a detailed report (default).

TEXT function key

**FRM_OFF (--)**
Frame (SDLC) layer data is not displayed.

OFF function key

**FRM_HEX (--)**
Displays frame (SDLC) layer data in hex.

HEX function key

**FRM_CHAR (--)**
Displays frame (SDLC) layer data in the currently selected character set.

CHARACTER function key
→ Transmission Header

TH_ON (--)  
Displays transmission header information in a detailed report (default).

กด TEXT function key

TH_OFF (--)  
Transmission header information is not displayed.

กด OFF function key

TH_HEX (--)  
Displays transmission header information in hex.

กด HEX function key

TH_CHAR (--)  
Displays transmission header information in the currently selected character set.

กด CHARACTER function key

→ Request/Resp Header

RH_ON (--)  
Displays request/response header information in a detailed report (default).

กด TEXT function key

RH_OFF (--)  
Request/response header information is not displayed.

กด OFF function key

RH_HEX (--)  
Displays request/response header information in hex.

กด HEX function key

RH_CHAR (--)  
Displays request/response header information in the currently selected character set.

กด CHARACTER function key
→ Sense Data

**SD_ON** (---)
Displays sense data information in a detailed report (default).

**TEXT** function key

**SD_OFF** (---)
Sense data information is not displayed.

**OFF** function key

**SD_HEX** (---)
Displays sense data information in hex.

**HEX** function key

**SD_CHAR** (---)
Displays sense data information in the currently selected character set.

**CHARACTER** function key

→ Request/Resp Unit

**RU_ON** (---)
Displays request/response unit information in a detailed report (default).

**TEXT** function key

**RU_OFF** (---)
Request/response unit information is not displayed.

**OFF** function key

**RU_HEX** (---)
Displays request/response unit information in hex.

**HEX** function key

**RU_CHAR** (---)
Displays request/response unit information in the currently selected character set.

**CHARACTER** function key
→ **Data Field**

**DATA_ON** or **DATA_CHAR** ( -- )
Displays the data field in the currently selected character set (default).


**DATA_OFF** ( -- )
The data field is not displayed.


**DATA_HEX** ( -- )
Displays the data field in hex format.


**CLEAR_CRT** ( -- )
Clears the display in the Data Window.


→ **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 7-1 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  AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+2</td>
<td>AP #1  AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+7</td>
<td>AP #1  AP #3</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+7</td>
<td>AP #2  AP #3</td>
</tr>
<tr>
<td>BRA/WAN</td>
<td>DUAL_3+4</td>
<td>AP #1  AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+2</td>
<td>AP #1  AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+3</td>
<td>AP #1  AP #4</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+4</td>
<td>AP #1  AP #5</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+7</td>
<td>AP #1  AP #3</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+3</td>
<td>AP #2  AP #4</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+4</td>
<td>AP #2  AP #5</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+7</td>
<td>AP #2  AP #3</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+4</td>
<td>AP #4  AP #5</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+7</td>
<td>AP #4  AP #3</td>
</tr>
<tr>
<td></td>
<td>DUAL_4+7</td>
<td>AP #5  AP #3</td>
</tr>
<tr>
<td>BRA/BRA</td>
<td>DUAL_1+2</td>
<td>AP #1  AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+3</td>
<td>AP #1  AP #4</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+4</td>
<td>AP #1  AP #5</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+5</td>
<td>AP #1  AP #6</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+7</td>
<td>AP #1  AP #3</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+3</td>
<td>AP #2  AP #4</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+4</td>
<td>AP #2  AP #5</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+5</td>
<td>AP #2  AP #6</td>
</tr>
<tr>
<td></td>
<td>DUAL_2+7</td>
<td>AP #2  AP #3</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+4</td>
<td>AP #4  AP #5</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+5</td>
<td>AP #4  AP #6</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+7</td>
<td>AP #4  AP #3</td>
</tr>
<tr>
<td></td>
<td>DUAL_4+5</td>
<td>AP #5  AP #6</td>
</tr>
<tr>
<td></td>
<td>DUAL_4+7</td>
<td>AP #5  AP #3</td>
</tr>
<tr>
<td></td>
<td>DUAL_5+7</td>
<td>AP #6  AP #3</td>
</tr>
<tr>
<td>PRA/WAN</td>
<td>DUAL_1+3</td>
<td>AP #1  AP #2</td>
</tr>
<tr>
<td></td>
<td>DUAL_1+4</td>
<td>AP #1  AP #3</td>
</tr>
<tr>
<td></td>
<td>DUAL_3+4</td>
<td>AP #2  AP #3</td>
</tr>
</tbody>
</table>

Table 7–1  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.

COMPLETE 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 SNA 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 DTE and DCE data).

NOTE
For accurate throughput measurement, the bit rate (line speed) must be set on the Monitor/Emulation Configuration Menu or in the INTERFACE-SPEED variable to match the actual line speed.

The baud rate, as stored in the INTERFACE-SPEED variable, is used to calculate a percentage throughput based on theoretical limits.

INTERFACE-SPEED ( -- )
Contains the current bit rate (default value is 64000).

Example:
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
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
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 frames, and activate or deactivate each of the filters.

<table>
<thead>
<tr>
<th>Filter Type</th>
<th>DISPLAY</th>
<th>Filter Status</th>
<th>ACTIVATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Statements</td>
<td>OFF</td>
<td>Selective PU</td>
<td>193 ALL</td>
</tr>
<tr>
<td>Lead Changes</td>
<td>BLOCK</td>
<td>Selective LU</td>
<td>ALL</td>
</tr>
</tbody>
</table>

Frame Layer:
- SNRM: BLOCK I PASS SIM BLOCK CFGR BLOCK
- DISC: BLOCK RR BLOCK RIM BLOCK FRMR BLOCK
- UA: BLOCK RNR BLOCK BCN BLOCK UP BLOCK
- DM: BLOCK REJ BLOCK XID BLOCK Invalid BLOCK
- RD: BLOCK UI BLOCK TEST BLOCK

Figure 8–1  Filter Setup Menu 1

→ 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 the filter status is changed, the connection diagram changes to reflect this. Figure 8–2 shows live data as the data source with display filters activated. If deactivated, all lead changes, trace statements, frames, transmission headers, request/response headers, and sense data go to the display.
Figure 8-2 Connection Diagram – Display Filters Activated

**REP_FILTER_ON** (---)
Activates the display filter.

- Filter Type
  - DISPLAY function key
- Filter Status
  - ACTIVATED function key

**REP_FILTER_OFF** (---)
Deactivates the display filter.

- Filter Type
  - DISPLAY function key
- Filter Status
  - DEACTIVATED function key

**RAM_FILTER_ON** (---)
Activates the capture RAM filter.

- Filter Type
  - RAM function key
- Filter Status
  - ACTIVATED function key
RAM_FILTER_OFF (---)
Deactivates the RAM filter.

→ Filter Type
   RAM function key
→ Filter Status
   DEACTIVATED function key

DISK_FILTER_ON (---)
Activates the disk filter.

→ Filter Type
   DISK function key
→ Filter Status
   ACTIVATED function key

DISK_FILTER_OFF (---)
Deactivates the disk filter.

→ Filter Type
   DISK function key
→ Filter Status
   DEACTIVATED function key

→ Trace Statements
Trace statements can be blocked or passed (default).

YES RTRACE (---)
Passes trace statements to the display.

→ Filter Type
   DISPLAY function key
→ Trace Statements
   ON function key

NO RTRACE (---)
Blocks trace statements from the display.

→ Filter Type
   DISPLAY function key
→ Trace Statements
   OFF function key

YES CTRACE (---)
Passes trace statements to capture RAM.

→ Filter Type
   RAM function key
→ Trace Statements
   ON function key
NO CTRACE ( -- )
Blocks trace statements from capture RAM.

→ Filter Type
   RAM function key
→ Trace Statements
   OFF function key

YES DTRACE ( -- )
Passes trace statements to disk.

→ Filter Type
   DISK function key
→ Trace Statements
   ON function key

NO DTRACE ( -- )
Blocks trace statements from disk.

→ Filter Type
   DISK function key
→ Trace Statements
   OFF function key

→ Lead Changes
Lead changes can be blocked (default) or passed.

R1=ALL ( -- )
Passes lead changes to the display.

→ Filter Type
   DISPLAY function key
→ Lead Changes
   PASS function key

R1=NONE ( -- )
Blocks lead changes from the display.

→ Filter Type
   DISPLAY function key
→ Lead Changes
   BLOCK function key

C1=ALL ( -- )
Passes lead changes to capture RAM.

→ Filter Type
   RAM function key
→ Lead Changes
   PASS function key
**C1=None (---)**
Blocks lead changes from capture RAM.

- **Filter Type**
  - RAM function key
- **Lead Changes**
  - BLOCK function key

**D1=ALL (---)**
Passes lead changes to disk.

- **Filter Type**
  - DISK function key
- **Lead Changes**
  - PASS function key

**D1=None (---)**
Blocks lead changes from the disk.

- **Filter Type**
  - DISK function key
- **Lead Changes**
  - BLOCK function key

→ **Selective PU**
The link station address PU (physical unit) is contained in the first byte of all received frames.

If one address has been selected and the display filter is activated, only frames with the specified value in the first byte are displayed unless blocked by another filter.

**RFRMA=ALL (---)**
Passes all PU's to the display unless blocked by another filter.

- **Filter Type**
  - DISPLAY function key
- **Selective PU**
  - ALL function key

**CFRMA=ALL (---)**
Passes all PU's to capture RAM unless blocked by another filter.

- **Filter Type**
  - RAM function key
- **Selective PU**
  - ALL function key
DFRMA=ALL ( -- )
Passes all PU's to disk unless blocked by another filter.

Filter Type
≡ DISK function key
≡ Selective PU

DO_RFRMA ( PU value\1 -- )
Specifies the PU address used for display filters. Once display filters are activated, only frames with this address field (first byte of frame) are displayed. Valid PU values are 0 (default) through 255. The '1' must always be included.

Example 1:
Display only traffic to/from physical unit 27.
27 1 DO_RFRMA ( Define selective PU )
REP_FILTER_ON ( Activate filter )

Example 2:
Display only traffic to/from physical unit X'ZZ'.
0xZZ 1 DO_RFRMA

NOTE
Filters can be set for a hexadecimal station address by prefixing with '0x'.

Filter Type
≡ DISPLAY function key
≡ Selective PU
≡ ONE function key

DO_CFRMA ( PU value\1 -- )
Specifies the PU address used for RAM filters. Once RAM filters are activated, only frames with this address field (first byte of frame) are captured. Valid PU values are 0 (default) through 255. The '1' must always be included.

Example:
Capture only traffic to/from physical unit 54.
54 1 DO_CFRMA ( Define selective PU )
RAM_FILTER_ON ( Activate filter )

Filter Type
≡ RAM function key
≡ Selective PU
≡ ONE function key
DO_DFRMA (PU value\1 -- )
Specifies the PU address used for disk filters. Once disk filters are activated, only frames with this value in the address field (first byte of frame) are recorded. Valid PU values are 0 (default) through 255. The ‘1’ must always be included.

→ Filter Type
   DISK function key
→ Selective PU
   ONE function key

Example:
Record only traffic to/from physical unit 100 to disk.
100 1 DO_DFRMA (Define selective PU)
DISK_FILTER_ON (Activate filter)

→ Selective LU
The LU (logical unit) is either the destination or origination address field in an FID0, FID1, or FID2 transmission header. This is not applicable for FID3, FID4 or FIDF.

⚠️ NOTE
The selective LU and selective PU work in an 'AND' condition (i.e. both the selective LU and PU must match in order to pass filters).

RTHAF=ALL
Passes all DAF and OAF (destination and origination) addresses to the display.

→ Filter Type
   DISPLAY function key
→ Selective LU
   ALL function key

CTHAF=ALL
Passes all DAF and OAF (destination and origination) addresses to capture RAM.

→ Filter Type
   RAM function key
→ Selective LU
   ALL function key

DTHAF=ALL
Passes all DAF and OAF (destination and origination) addresses to disk.

→ Filter Type
   DISK function key
→ Selective LU
   ALL function key
DO_RTHAF ( RU value/1 -- )
Specifies the selective LU address field for display filters. Once display filters are activated, only FID's (FIDO, FID1, or FID2) with this value in either the destination or origination address field are displayed. Valid values are 0 through 65535 for FIDO and FID1, and 0 through 255 for FID2. The '1' must always be included.

Example:
Set a display filter for a selective LU to a value of 75.
75 1 DO_RTHAF ( Define selective LU address )
REP_FILTER_ON ( Activate display filters )

→ Filter Type
  DISPLAY function key
→ Selective LU
  ONE function key

DO_CTHAF ( LU value/1 -- )
Specifies the selective LU address field for RAM filters. Once RAM filters are activated, only FID's (FIDO, FID1, or FID2) with this value in either the destination or origination address field are captured to RAM. Valid values are 0 through 65535 for FIDO and FID1, and 0 through 255 for FID2. The '1' must always be included.

Example:
Set a RAM filter for a selective LU to 100.
100 1 DO_CTHAF ( Define selective LU address )
RAM_FILTER_ON ( Activate RAM filter )

→ Filter Type
  RAM function key
→ Selective LU
  ONE function key

DO_DTHAF ( LU value/1 -- )
Specifies the selective LU address field for disk filters. Once disk filters are activated, only FID's (FIDO, FID1, or FID2) with this value in either the destination or origination address field are recorded to disk. Valid values are 0 through 65535 for FIDO and FID1, and 0 through 255 for FID2. The '1' must always be included.

Example:
Set a disk filter for a selective LU to 2.
2 1 DO_DTHAF ( Define selective LU address )
DISK_FILTER_ON ( Activate RAM filter )

→ Filter Type
  DISK function key
→ Selective LU
  ONE function key
Frame Layer:
If display filters are activated, any frame with a PASS status is displayed unless the selective PU address has been set to a specific address. See the description under the Selective PU Address.

\[\text{NOTE}\]
If the I frame has been set to a BLOCK status, a filter at higher layers is inappropriate. This is indicated by the dashes shown beside the individual items.

\[\text{NOTE}\]
Commands for display filters and SNRM frames are described here as an example. For a complete list of commands, see Table 8-1.

RFRM+SNRM
Passes SNRM frames to the display.

\[\rightarrow \text{Filter Type}\]
\[\rightarrow \text{DISPLAY function key}\]
\[\rightarrow \text{SNRM}\]
\[\rightarrow \text{PASS function key}\]

RFRM–SNRM
Blocks SNRM frames from the display.

\[\rightarrow \text{Filter Type}\]
\[\rightarrow \text{DISPLAY function key}\]
\[\rightarrow \text{SNRM}\]
\[\rightarrow \text{BLOCK function key}\]

RFRM=ALL
Passes all frames (default) to the display.

\[\rightarrow \text{Filter Type}\]
\[\rightarrow \text{DISPLAY function key}\]
\[\rightarrow \text{SNRM}\]
\[\rightarrow \text{ALL FRAMES function key}\]

RFRM=NONE
Blocks all frames and packets from the display.

\[\rightarrow \text{Filter Type}\]
\[\rightarrow \text{DISPLAY function key}\]
\[\rightarrow \text{SNRM}\]
\[\rightarrow \text{NO FRAMES function key}\]
<table>
<thead>
<tr>
<th>Description</th>
<th>Display</th>
<th>RAM</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Pass RFRM=ALL CFRM=ALL DFRM=ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM=NONE CFRM=NONE DFRM=NONE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Normal Response Mode</td>
<td>Pass RFRM+SNRM CFRM+SNRM DFRM+SNRM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−SNRM CFRM−SNRM DFRM−SNRM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disconnect</td>
<td>Pass RFRM+DISC CFRM+DISC DFRM+DISC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−DISC CFRM−DISC DFRM−DISC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unnumbered Acknowledgment</td>
<td>Pass RFRM+UA CFRM+UA DFRM+UA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−UA CFRM−UA DFRM−UA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disconnect Mode</td>
<td>Pass RFRM+DM CFRM+DM DFRM+DM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−DM CFRM−DM DFRM−DM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame Reject</td>
<td>Pass RFRM+FRMR CFRM+FRMR DFRM+FRMR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−FRMR CFRM−FRMR DFRM−FRMR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receiver Ready</td>
<td>Pass RFRM+RR CFRM+RR DFRM+RR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−RR CFRM−RR DFRM−RR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receiver Not Ready</td>
<td>Pass RFRM+RNR CFRM+RNR DFRM+RNR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−RNR CFRM−RNR DFRM−RNR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reject</td>
<td>Pass RFRM+REJ CFRM+REJ DFRM+REJ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−REJ CFRM−REJ DFRM−REJ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Initialization Mode</td>
<td>Pass RFRM+SIM CFRM+SIM DFRM+SIM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−SIM CFRM−SIM DFRM−SIM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request Initialization Mode</td>
<td>Pass RFRM+RIM CFRM+RIM DFRM+RIM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−RIM CFRM−RIM DFRM−RIM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beacon</td>
<td>Pass RFRM+BCN CFRM+BCN DFRM+BCN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−BCN CFRM−BCN DFRM−BCN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configure</td>
<td>Pass RFRM+CFGR CFRM+CFGR DFRM+CFGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−CFGR CFRM−CFGR DFRM−CFGR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request Disconnect</td>
<td>Pass RFRM+RD CFRM+RD DFRM+RD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−RD CFRM−RD DFRM−RD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exchange Station Identification</td>
<td>Pass RFRM+XID CFRM+XID DFRM+XID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−XID CFRM−XID DFRM−XID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unnumbered Poll</td>
<td>Pass RFRM+UP CFRM+UP DFRM+UP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−UP CFRM−UP DFRM−UP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>Pass RFRM+TEST CFRM+TEST DFRM+TEST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−TEST CFRM−TEST DFRM−TEST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unnumbered Information</td>
<td>Pass RFRM+UI CFRM+UI DFRM+UI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−UI CFRM−UI DFRM−UI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td>Pass RFRM+I CFRM+I DFRM+I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−I CFRM−I DFRM−I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invalid</td>
<td>Pass RFRM+INV CFRM+INV DFRM+INV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RFRM−INV CFRM−INV DFRM−INV</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 8-1 Frame Layer Filter Commands**
Figure 8-3 Filter Setup Menu 2

Transmission Header:
If display filters are activated, any FID with a PASS status is displayed unless the selective PU address or the selective LU has been set to a specific value. See the descriptions under Selective PU and Selective LU.

⚠️ NOTE
Commands for display filters and FIDO transmission headers are described here as an example. For a complete list of commands, see Table 8-2.

RFID+0
Passes FIDO transmission headers to the display.

➡️ Filter Type
  DISPLAY function key
  FIDO
  PASS function key

RFID−0
Blocks FIDO transmission headers from the display.

➡️ Filter Type
  DISPLAY function key
  FIDO
  BLOCK function key
RFID=ALL
Passes all transmission headers to the display.

* Filter Type
  DISPLAY function key
  FID0
  ALL FID function key

RFID=None
Blocks all transmission headers from the display.

* Filter Type
  DISPLAY function key
  FID0
  NO FID 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</td>
<td>RFID=ALL</td>
<td>CFID=ALL</td>
</tr>
<tr>
<td>Block</td>
<td>Block</td>
<td>RFID=NONE</td>
<td>CFID=NONE</td>
</tr>
<tr>
<td>TH FID type = 0</td>
<td>Pass</td>
<td>RFID+0</td>
<td>CFID+0</td>
</tr>
<tr>
<td>Block</td>
<td>Block</td>
<td>RFID-0</td>
<td>CFID-0</td>
</tr>
<tr>
<td>TH FID type = 1</td>
<td>Pass</td>
<td>RFID+1</td>
<td>CFID+1</td>
</tr>
<tr>
<td>Block</td>
<td>Block</td>
<td>RFID-1</td>
<td>CFID-1</td>
</tr>
<tr>
<td>TH FID type = 2</td>
<td>Pass</td>
<td>RFID+2</td>
<td>CFID+2</td>
</tr>
<tr>
<td>Block</td>
<td>Block</td>
<td>RFID-2</td>
<td>CFID-2</td>
</tr>
<tr>
<td>TH FID type = 3</td>
<td>Pass</td>
<td>RFID+3</td>
<td>CFID+3</td>
</tr>
<tr>
<td>Block</td>
<td>Block</td>
<td>RFID-3</td>
<td>CFID-3</td>
</tr>
<tr>
<td>TH FID type = 4</td>
<td>Pass</td>
<td>RFID+4</td>
<td>CFID+4</td>
</tr>
<tr>
<td>Block</td>
<td>Block</td>
<td>RFID-4</td>
<td>CFID-4</td>
</tr>
<tr>
<td>TH FID type = F</td>
<td>Pass</td>
<td>RFID+F</td>
<td>CFID+F</td>
</tr>
<tr>
<td>Block</td>
<td>Block</td>
<td>RFID-F</td>
<td>CFID-F</td>
</tr>
</tbody>
</table>

Table 8-2 Transmission Header Filter Commands

Request/Response Header:
If display filters are activated, any request/response header with a PASS status is displayed unless the selective PU address or the selective LU has been set to a specific value. See the description under the Selective Address PU and Selective LU.

* NOTE
Commands for display filters and FMD request/response headers are described here as an example. For a complete list of commands, see Table 8-3.
FILTERS

RRH+FMD
Passes FMD request/response headers to the display.

→ Filter Type
  DISPLAY function key
  → FMD
    PASS function key

RRH−FMD
Blocks FMD request/response headers from the display.

→ Filter Type
  DISPLAY function key
  → FMD
    BLOCK function key

RRH=ALL
Passes all request/response headers to the display.

→ Filter Type
  DISPLAY function key
  → FMD
    ALL REQ/RES function key

RRH=NONE
Blocks all request/response headers from the display.

→ Filter Type
  DISPLAY function key
  → FMD
    NO REQ/RES 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</td>
<td>RRH−ALL</td>
<td>CRH−ALL</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RRH−NONE</td>
<td>CRH−NONE</td>
</tr>
<tr>
<td>Function Management Data RH</td>
<td>Pass</td>
<td>RRH+FMD</td>
<td>CRH+FMD</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RRH−FMD</td>
<td>CRH−FMD</td>
</tr>
<tr>
<td>Network Control RH</td>
<td>Pass</td>
<td>RRH−NC</td>
<td>CRH−NC</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RRH−NC</td>
<td>CRH−NC</td>
</tr>
<tr>
<td>Data Flow Control RH</td>
<td>Pass</td>
<td>RRH+DFC</td>
<td>CRH+DFC</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RRH−DFC</td>
<td>CRH−DFC</td>
</tr>
<tr>
<td>Session Control RH</td>
<td>Pass</td>
<td>RRH+SC</td>
<td>CRH+SC</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RRH−SC</td>
<td>CRH−SC</td>
</tr>
<tr>
<td>Invalid RH</td>
<td>Pass</td>
<td>RRH+INV</td>
<td>CRH+INV</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RRH−INV</td>
<td>CRH−INV</td>
</tr>
</tbody>
</table>

Table 8-3 Request/Response Header Filter Commands
Sense Data:
If display filters are activated, any sense data with a PASS status is displayed unless the selective PU or selective LU has been set to a specific value. See the description under the Selective Address and Selective LU.

⚠️ NOTE
Commands for display filters and path error sense data are described here as an example. For a complete list of commands, see Table 8-4.

**RSD+PER**
Passes path error sense data to the display.

![Filter Type]
- DISPLAY function key
- PATH ERROR
  - PASS function key

**RSD-PER**
Blocks path error sense data from the display.

![Filter Type]
- DISPLAY function key
- PATH ERROR
  - BLOCK function key

**RSD=ALL**
Passes all sense data to the display.

![Filter Type]
- DISPLAY function key
- PATH ERROR
  - ALL SD function key

**RSD=NONE**
Blocks all sense data from the display.

![Filter Type]
- DISPLAY function key
- PATH ERROR
  - NO SD function key

⚠️ NOTE
A frame containing both request/response header and sense data can only be blocked if both are blocked.
<table>
<thead>
<tr>
<th>Description</th>
<th>Display</th>
<th>RAM</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Pass</td>
<td>RSD=ALL</td>
<td>CSD=ALL</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RSD=NONE</td>
<td>CSD=NONE</td>
</tr>
<tr>
<td>Request Reject</td>
<td>Pass</td>
<td>RSD+REQRJ</td>
<td>CSD+REQRJ</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RSD−REQRJ</td>
<td>CSD−REQRJ</td>
</tr>
<tr>
<td>User</td>
<td>Pass</td>
<td>RSD+USER</td>
<td>CSD+USER</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RSD−USER</td>
<td>CSD−USER</td>
</tr>
<tr>
<td>Request Error</td>
<td>Pass</td>
<td>RSD+REQER</td>
<td>CSD+REQER</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RSD−REQER</td>
<td>CSD−REQER</td>
</tr>
<tr>
<td>State Error</td>
<td>Pass</td>
<td>RSD+STER</td>
<td>CSD+STER</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RSD−STER</td>
<td>CSD−STER</td>
</tr>
<tr>
<td>RH Usage Error</td>
<td>Pass</td>
<td>RSD+RHER</td>
<td>CSD+RHER</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RSD−RHER</td>
<td>CSD−RHER</td>
</tr>
<tr>
<td>Path Error</td>
<td>Pass</td>
<td>RSD+PER</td>
<td>CSD+PER</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RSD−PER</td>
<td>CSD−PER</td>
</tr>
<tr>
<td>Invalid</td>
<td>Pass</td>
<td>RSD+INV</td>
<td>CSD+INV</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>RSD−INV</td>
<td>CSD−INV</td>
</tr>
</tbody>
</table>

Table 8–4 Sense Data Filter Commands
This section displays the Emulation Configuration Menu and describes commands corresponding to each item.

<table>
<thead>
<tr>
<th>Emulation Configuration Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>➔ Emulation Mode</td>
</tr>
</tbody>
</table>

**Physical Layer:**
- **Emulation Interface:** TO DCE
- **Interface Type:** RS232C/V.28
- **Interface Leads:** DISABLED

**Frame Layer:**
- **Emulation:** AUTOMATIC
- **Max Frame Size:** 4096
- **Secondary Stations:** 32
- **Bit Rate:** UNKNOWN
- **Clocking:** NRZ WITH CLOCK
- **Interframe Fill:** FLAG
- **Multipoint:** ALTERNATE
- **Two Way Mode:** ALTERNATE
- **Poll Timer (Sec):** 1.0

*Figure 9-1 Emulation Configuration Menu*

**STARTUP (---)**
Initializes the SDLC Emulation and configures the physical layer.

**NOTE**  
*Use STARTUP once after all physical layer changes are made.*

### 9.1 Emulation Mode

**PRIMARY (---)**
Emulates a primary SDLC device and starts the primary polling timer. Selects the primary state machine and forces the emulation into the NO_LINK state (see Table 12-1).

*PRIMARY function key*

**SECONDARY (---)**
Emulates a secondary SDLC device. Selects the secondary state machine (default) and forces the emulation into the NDM state (see Table 12-2).

*SECONDARY function key*
9.2 Physical Layer

The emulation is active at all times. i.e. automatic responses to protocol events are generated even during parameter setup.

→ Emulation Interface
Selects the physical type of emulation.

⚠️ NOTE
Refer to Table 9–1 for clocking selections depending on the emulation interface.

TO_DCE_IF ( -- )
Selects the 'to DCE' interface (default).

TO DCE function key

TO_DTE_IF ( -- )
Selects the 'to DTE' interface.

TO DTE function key

→ Interface Type

IF=V28 ( -- )
Selects the V.28/RS-232C connector (default) and electrically isolates the other connectors on the port.

RS232C/V.28 function key

IF=V11 ( -- )
Selects the V.11/X.21 connector for protocol testing and electrically isolates the other connectors on the port.

RS422/V.11 function key

IF=V35 ( -- )
Selects the V.35 connector for protocol testing and electrically isolates the other connectors on the port.

V.35 function key

IF=V36 ( -- )
Selects the V.36/RS-449 connector for protocol testing 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.
→ **Interface Leads**

Individual or all interface leads can be enabled or disabled (default). Leads must be enabled for test manager detection.

**ENABLE_LEAD** (lead identifier -- )

Enables the specified lead. Refer to the Programmer's Reference Manual for a list of supported leads for each interface type.

Example:
Enable the request to send lead.

IRS ENABLE_LEAD

**DISABLE_LEAD** (lead identifier -- )

Disables (default) the specified lead. Refer to the Programmer's Reference Manual for a list of supported leads for each interface type.

Example:
Disable the clear to send lead.

ICS DISABLE_LEAD

**ALL_LEADS** (-- lead identifier)

Enables/disables all leads supported on the currently selected WAN interface. **ALL_LEADS** must be used with **ENABLE_LEAD** or **DISABLE_LEAD**.

Example 1:
Enable all leads for the current interface.

ALL_LEADS ENABLE_LEAD

[ ENABLE ] function key

Example 2:
Disable all leads for the current interface.

ALL_LEADS DISABLE_LEAD

[ DISABLE ] function key
→ Bit Rate
The interface speed can be selected from preset values on the Interface Port Speed Menu, set to a user-defined speed, or measured 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>NRZ WITH CLOCK</td>
<td>Select</td>
<td>Measure</td>
</tr>
<tr>
<td>EXTERNAL TX CLOCK</td>
<td>Measure</td>
<td>Select</td>
</tr>
<tr>
<td>NRZI</td>
<td>Select</td>
<td>Select</td>
</tr>
<tr>
<td>NRZI WITH CLOCK</td>
<td>Select</td>
<td>Measure</td>
</tr>
</tbody>
</table>

Table 9-1 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.

The following commands can be used whenever the bit rate can be selected.

```
SPEED=50  SPEED=300  SPEED=4800  SPEED=38400
SPEED=75  SPEED=1200 SPEED=7200  SPEED=48000
SPEED=110 SPEED=1800 SPEED=9600 (default)  SPEED=56000
SPEED=134.5 SPEED=2000 SPEED=14400  SPEED=64000
SPEED=150  SPEED=2400 SPEED=16000  SPEED=72000
SPEED=200  SPEED=3600 SPEED=19200  SPEED=128000
```

Example:
Set the bit rate of the emulation to 64000.
SPEED=64000

`INTERFACE–SPEED (--address)`
Contains the current bit rate (default is 9600).

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`.

→ Clocking

**IF=NRZ (--)**
Expects standard non-return to zero line encoding (default) with DCE provided clocks.

 NRZ WITH CLOCK function key

**IF=NRZ_EXT (--)**
Selects a DTE provided transmit clock on pint 24 of an RS–232C connector.

 INTERNAL TX CLOCK function key
IF=NRZI (--) 
Selects the non-return to zero inverted method of encoding with timing information extracted from the data signal.

NRZI function key

IF=NRZI+CLK (--) 
Selects the non-return to zero inverted method of encoding with timing information extracted from the provided clock signal.

NRZI WITH CLOCK function key

NOTE
If any of these commands are used when the application is running on a B-Channel, the clocking is forced to IF=NRZ.

Interframe Fill
Selects the bit pattern transmitted between blocks of data.

IDLE_MARK (--) 
Sends a mark condition during the interframe idle times.

MARK function key

IDLE_FLAG (--) 
Sends flags (hex 7E) during the interframe idle times (default).

FLAG function key

The following sequence illustrates the use of the configuration commands. STARTUP is only called at the end of the configuration sequences.

IF=V35 ( Use V.35 test connector )
PRIMARY ( Emulate a primary SDLC device )
TO_DCE_IF ( Set for full DTE simulation )
56000 INTERFACE-SPEED ! ( Set baud rate )
IF=NRZ ( Clocking provided by DCE )
STARTUP ( Configure software )

SDLC/SNA Programmer's Manual
9.3 Frame Layer

IDACOM's SDLC Emulation implements an automatic layer 2 state machine.

→ Emulation

**EMUL_OFF** ( -- )
Disables the automatic emulation; the SDLC state machine does not transmit any frames without manual or test program intervention.

\[ MANUAL \] function key

**EMUL_ON** ( -- )
Enables the automatic emulation; the SDLC state machine responds to all incoming frames according to SDLC protocol.

\[ AUTOMATIC \] function key

→ Max Frame Size

**MAX-FRAME-LENGTH** ( -- address )
Contains the maximum frame size. Valid values are 1 through 4096 (default).

Example:
Set the maximum frame size to 2000.

2000  MAX-FRAME-LENGTH !

→ Secondary Stations

**DO_#LINKS** (number of secondaries\1--)
Specifies the number of active secondary stations that can be emulated simultaneously. Valid values are 1 through 32 (default). The ‘1’ must always be included.

Example:
Set the number of secondary stations to 20.

20 1 DO_#LINKS

A safer method of setting the number of secondary stations to 20 is:

20 DUP 1 32 BETWEEN? \( \text{ (Check to see if value is between 1 and 32 )} \)

IF
1 DO_#LINKS
ELSE
DROP
ENDIF

\[ Modify Number of Active Secondary Stations \] function key
→ **Multipoint**

**PT_TO_PT (--)**
Selects point-to-point configuration (default). In this mode, only one secondary can transmit a batch of messages at a time.

- **OFF** function key

**MULTI_PT (--)**
Selects multipoint configuration. All the links can have active transmissions at the same time.

- **ON** function key

→ **Two Way Mode**

**TWA (--)**
Uses two way alternate mode (default) for all transmissions (also referred to as half duplex mode).

- **ALTERNATE** function key

**TWS (--)**
Uses two way simultaneous mode for all transmissions (also referred to as full duplex mode).

- **SIMULTANEOUS** function key

→ **Poll Timer (Sec)**

**DO_PTIME** (value1 --)
Specifies the timeout, in tenths of seconds, of the poll timer for primary emulation (default is 1 second).

If the value is 0, the poll timer is forced to one-tenth of a second and is then restarted. The '1' must always be included.

Example:
Set the poll timer to 2 seconds.
20 1 DO_PTIME

- **Modify Primary Poll Timer** function key
9.4 Station Setup

Individual data links can be configured via the Station Setup Menu. The SDLC Emulation supports 32 stations.

<table>
<thead>
<tr>
<th>Secondary Station</th>
<th>Address</th>
<th>Window</th>
<th>Timeout(Sec)</th>
<th>Retries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secondary 0</td>
<td>1</td>
<td>4</td>
<td>3.0</td>
<td>20</td>
</tr>
<tr>
<td>Secondary 1</td>
<td>2</td>
<td>4</td>
<td>3.0</td>
<td>20</td>
</tr>
<tr>
<td>Secondary 2</td>
<td>3</td>
<td>4</td>
<td>3.0</td>
<td>20</td>
</tr>
<tr>
<td>Secondary 3</td>
<td>4</td>
<td>4</td>
<td>3.0</td>
<td>20</td>
</tr>
<tr>
<td>Secondary 4</td>
<td>5</td>
<td>4</td>
<td>3.0</td>
<td>20</td>
</tr>
<tr>
<td>Secondary 5</td>
<td>6</td>
<td>4</td>
<td>3.0</td>
<td>20</td>
</tr>
<tr>
<td>Secondary 6</td>
<td>7</td>
<td>4</td>
<td>3.0</td>
<td>20</td>
</tr>
<tr>
<td>Secondary 7</td>
<td>8</td>
<td>4</td>
<td>3.0</td>
<td>20</td>
</tr>
<tr>
<td>Secondary 8</td>
<td>9</td>
<td>4</td>
<td>3.0</td>
<td>20</td>
</tr>
<tr>
<td>Secondary 9</td>
<td>A</td>
<td>4</td>
<td>3.0</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 9-2 Station Setup Menu

The number of supported secondary stations is specified with the DO_#LINKS command.

NOTE
The commands for selecting a station and specifying the station, address and window size can be used in both primary and secondary emulations. The commands for specifying the timeout and retries apply only to a primary emulation.

DO_LINK# (station number\1--)
Specifies the secondary station for configuration. Valid values are 0 through 31. If the specified station number is greater than the actual number supported, the emulation selects the last station supported. The ‘1’ must always be included.

Example:
Select secondary station 30.
30 1 DO_LINK#
EMULATION CONFIGURATION

SET_LINK_ID (station#\station address -- )
Specifies the address of the secondary station. Valid values are 1 through 254.

If the specified station number is greater than the actual number supported, the emulation
selects the last station supported.

Example:
Set the address for secondary station 20 to hex 10.
20 0X10 SET_LINK_ID

Address function key

DO_SETW (window size\1-- )
Specifies the window size (number of frames sent before an acknowledgment is received) of
the currently selected secondary station. Valid values are 1 through 7. The '1' must always
be included.

Example:
Set the window size on station 0 to 5.
0 1 DO_LINK# (Select station 0 )
5 1 DO_SETW (Set the window size to 5 )

Window function key

DO_SETTO (tenths of seconds/1-- )
Specifies the timeout, in tenths of seconds, the primary waits before retrying a secondary after
receiving no response (default is 3 seconds). The '1' must always be included.

Example:
Set the time interval between retries for station number 6 to 5 seconds.
6 1 DO_LINK# (Select station 6 )
50 1 DO_SETTO (Set the time interval to 5 seconds )

Timeout function key

DO_SETRETRY (#of retries\1-- )
Specifies the number of times the primary polls a selected secondary after receiving no
response (default is 20 retries). The '1' must always be included.

Example:
Set the number of retries to 10 for station 4.
4 1 DO_LINK# (Select station 4 )
10 1 DO_SETRETRY (Set number of retries to 10 )

Retries function key
This section describes the structure of the SDLC Emulation. The IDACOM SDLC Emulation program is a combination of the complete SNA monitor application package together with an SDLC state machine. All commands available in the SNA Monitor are also available in the SDLC Emulation. The program's data flow is detailed for the reception of protocol events (frames, lead changes or timers) and generating responses to these events.

Received frames are first decoded in the emulation decode block and then passed on to the test manager if a test script is running. The test manager can generate data, start timers, etc., in response to the received frame, or strictly recognize that a particular frame has been received with no associated output. After the test manager has processed the received frame, the SDLC state machine processes the received frame and generates any necessary protocol responses. By handling the received frame in this sequence, the automatic state machine operation can be turned off via the test manager, prior to running the SDLC state machine.

10.1 Live Data

The SNA Monitor decodes any received/transmitted frames and displays them for user interpretation while the SDLC state machine interprets the received frames and forces some action (usually transmitting a frame, RR, RNR, etc.).
The emulation receives events from the interface and processes them as shown in Figure 10-1.

By default, the SDLC Emulation captures the received/transmitted data in the capture RAM buffer and displays it on the screen in a short format report. The SDLC Emulation is running (active) and responds to all SDLC layer frames. The emulation can be enabled or disabled with the EMUL_ON or EMUL_OFF commands.

### 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.

**EMUL_ON** (---)
Enables the automatic emulation; the SDLC state machine responds to all incoming frames according to the SDLC protocol.

**EMUL_OFF** (---)
Disables the automatic emulation; the SDLC state machine does not transmit any frames without manual or test program intervention.
10.2 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 10-2.

![SDLC Emulation Data Flow Diagram - Offline Processing](image)

**Figure 10-2** SDLC 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 the capture RAM or disk file and decoded and displayed or printed. RAM capture is suspended in this mode.
10.3 Simultaneous Live Data and Playback

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

Figure 10-3  SDLC 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.
This section describes the data flow diagram for the emulation decode and lists the variables in which decoded information is saved.

The SDLC Emulation is capable of modulo 8 decoding and response only. The standards followed have been defined by IBM.

![Data Link Layer Decode](image)

*Figure 11-1 SDLC Emulation Data Flow Diagram – Decode*

### 11.1 Communications 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 IBM's publication GA27-3093-2, IBM Synchronous Data Link Control, General Information. 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.*
Data Link Receive

ID (address)
Contains the address byte of the last received frame. The contents of this variable must be obtained by using C@. Valid values are 0 through 255.

CONTROL (address)
Contains the control byte of the last received frame. The contents of this variable must be obtained by using C@. Valid values are shown in Tables A-1 and A-2.

L2-FRAME-POINTER (address)
Contains the pointer to the received frame.

Example:
Obtain the control byte of the last received frame (there are three methods).
L2-FRAME-POINTER @ 1+ C@ (From the frame pointer)
or,
CONTROL C@ (From the emulation decode)
or,
M-CONTROL @ (From the monitor decode)

NR (address)
Contains the N(R) (received sequence count) of the last received supervisory or information frame. Valid values are 0 through 7. The SDLC Emulation performs checking procedures to determine the appropriate action.

NS (address)
Contains the N(S) (send sequence count) of the last received Information frame. Valid values are 0 through 7. The SDLC Emulation performs checking procedures to determine the appropriate action.

Data Link Transmit

LINK# (address)
Contains the currently selected link for frame transmission. Valid values are 0 through 31.

#SECS (address)
Contains the number of secondary stations supported by the emulation. Valid values are 1 through 32.

ID-OUT (address)
Contains the current PU address. This address is used in the ‘S:’ commands for transmitting frames (see Section 12). Valid values are 0 through 255.

BYTE-COUNT (address)
Contains the length of the frame to be transmitted.
P/F_BIT (-- address )
Contains the P/F bit used when transmitting frames with the 'S:' commands (see Section 12).
Valid values are 0 and 1 (default).

Transmission Header Receive

The following variables contain decoded information for received SNA information.

RDAF (-- address )
Contains the DAF (destination address field) located in bytes 2 and 3 of FID0 and FID1, and byte 2 of FID2. The field specifies which node the message is directed. Valid values are 0 through 65535 for FID0 and FID1, and 0 through 255 for FID2.

ROAF (-- address )
Contains the OAF (origination address field) located in bytes 4 and 5 of FID0 and FID1, and byte 3 of FID2. This field specifies the node transmitting the message. Valid values are 0 through 65535 for FID0 and FID1, and 0 through 255 for FID2.

RSNF (-- address )
Contains the SNF (sequence number field) located in bytes 6 and 7 of FID0 and FID1, bytes 4 and 5 of FID2, and bytes 22 and 23 of FID4. This field is generated by the LU (logical unit) and used for detection of lost messages. Valid values are 0 through 65535.

Transmission Header Transmit

S.FID-ID (-- address )
Contains the format identifier field of the FID2. As the BUILD_TH-BYTE0 or BYTE_TH commands are only supported for an FID2 header, the only permissible value for this variable is hex 20.

S.MAPPING-FIELD (-- address )
Contains the mapping field of the FID2.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*WHOLE-BIU (default)</td>
<td>Entire segment</td>
</tr>
<tr>
<td>R*FIRST-SEGMENT</td>
<td>First segment</td>
</tr>
<tr>
<td>R*LAST-SEGMENT</td>
<td>Last segment</td>
</tr>
<tr>
<td>R*MIDDLE-SEGMENT</td>
<td>Middle segment</td>
</tr>
</tbody>
</table>
S. EXP—FLOW—IND (— address)
Contains the expedited flow indicator field of the FID2.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*NONE</td>
<td>Normal</td>
</tr>
<tr>
<td>R*EXPEDITED</td>
<td>Expedited</td>
</tr>
</tbody>
</table>

SOAF (— address)
Contains the OAF (origin address field) of the FID2. Valid values are 0 (default) through 255.

SDAF (— address)
Contains the DAF (destination address field) of the FID2. Valid values are 0 (default) through 255.

SSNF (— address)
Contains the SNF (sequence number field) of the FID2 when the format indicator of the request/response header, within the same frame, is set to 0. Valid values are 0 (default) through 65535.

SSNF1 (— address)
Contains the SNF (sequence number field) of the FID2 when the format indicator of the request/response header, within the same frame, is set to 1. Valid values are 0 (default) to 65535.

Request/Response Header Transmit

S.RH—ID (— address)
Contains the RU category field of the request/response header.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>FMD (function management data)</td>
</tr>
<tr>
<td>20</td>
<td>NC (network control — default)</td>
</tr>
<tr>
<td>40</td>
<td>DFC (data flow control)</td>
</tr>
<tr>
<td>60</td>
<td>SC (session control)</td>
</tr>
</tbody>
</table>

S.RH—REQ—RES (— address)
Contains the request/response indicator.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Request header (default)</td>
</tr>
<tr>
<td>80</td>
<td>Response header</td>
</tr>
</tbody>
</table>
S.FORMAT-IND ( -- address )
Contains the format indicator field of the request/response header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>LU-LU Session Description</th>
<th>SSCP Session Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*FORMAT0</td>
<td>No FM header follows</td>
<td>Character coded RU (request/response unit)</td>
</tr>
<tr>
<td>R*FORMAT1 (default)</td>
<td>FM header follows</td>
<td>Field formatted RU</td>
</tr>
</tbody>
</table>

S.SENSE-DATA-IND ( -- address )
Contains the sense data indicator field of the request/response header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*NONE (default)</td>
<td>Sense data not included</td>
</tr>
<tr>
<td>R*SENSE-DATA-INCL</td>
<td>Sense data include (i.e. an error has occurred)</td>
</tr>
</tbody>
</table>

S.CHAIN-IND ( -- address )
Contains the begin and end chain fields of the request/response header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*MIDDLE-RU</td>
<td>Middle RU in the chain</td>
</tr>
<tr>
<td>R*LAST-RU</td>
<td>Last RU in the chain</td>
</tr>
<tr>
<td>R*FIRST-RU</td>
<td>First RU in the chain</td>
</tr>
<tr>
<td>R*ONLY-RU (default)</td>
<td>Only RU in the chain</td>
</tr>
</tbody>
</table>

S.RESPONSE-TYPE ( -- address )
Contains the DR1I, DR2I (definite response indicators 1 and 2), and the ER1 (exception response indicator) for a request header. The definite response indicators 1 and 2 and the response type indicator for a response header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Request Header</th>
<th>Response Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*RQN</td>
<td>No response requested</td>
<td>Reserved</td>
</tr>
<tr>
<td>R*RQD2</td>
<td>Definite response requested by DR2I</td>
<td>Positive response to DR2I</td>
</tr>
<tr>
<td>R*RQE2</td>
<td>Exception response requested by DR2I</td>
<td>Negative response to DR2I</td>
</tr>
<tr>
<td>R*RQD1 (default)</td>
<td>Definite response requested by DR1I</td>
<td>Positive response to DR1I</td>
</tr>
<tr>
<td>R*RQE1</td>
<td>Exception response requested by DR1I</td>
<td>Negative response to DR1I</td>
</tr>
<tr>
<td>R*RQD3</td>
<td>Definite response requested by DR1I and DR2I</td>
<td>Positive response to DR1I and DR2I</td>
</tr>
<tr>
<td>R*RQE3</td>
<td>Exception response requested by DR1I and DR2I</td>
<td>Negative response to DR1I and DR2I</td>
</tr>
</tbody>
</table>
S.QUEUED-RESP-IND (– address)
Contains the queued response indicator of the request/response header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*NONE (default)</td>
<td>Response bypasses transmission control queues</td>
</tr>
<tr>
<td>R*QUEUED</td>
<td>Enqueue response in transmission control queues</td>
</tr>
</tbody>
</table>

S.PACING-IND (– address)
Contains the pacing indicator field of the request/response header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*NONE (default)</td>
<td>Pacing not requested</td>
</tr>
<tr>
<td>R*PACING</td>
<td>Pacing requested</td>
</tr>
</tbody>
</table>

S.BRACKET-IND (– address)
Contains the begin and end bracket indicators of a request header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*NONE (default)</td>
<td>Between or within brackets</td>
</tr>
<tr>
<td>R*END</td>
<td>Notification to other half session that the speaker is finished and that the bracket protocol is complete</td>
</tr>
<tr>
<td>R*BEGIN</td>
<td>Request by a half session to become speaker of the session</td>
</tr>
<tr>
<td></td>
<td>Only chain in the bracket</td>
</tr>
</tbody>
</table>

S.CHANGE-DIR-IND (– address)
Contains the change direction indicator of the request header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*NONE (default)</td>
<td>No change in direction</td>
</tr>
<tr>
<td>R*CHANGE</td>
<td>Change direction</td>
</tr>
</tbody>
</table>

S.CODE-SEL-IND (– address)
Contains the code selection indicator of the request header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*CODE0 (default)</td>
<td>Code 0</td>
</tr>
<tr>
<td>R*CODE1</td>
<td>Code 1</td>
</tr>
</tbody>
</table>
S.ENCIPH–DATA–IND (-- address)
Contains the enciphered data indicator of the request header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*NONE (default)</td>
<td>RU not enciphered</td>
</tr>
<tr>
<td>R*PADDED</td>
<td>RU enciphered</td>
</tr>
</tbody>
</table>

S.PADDED–DATA–IND (-- address)
Contains the padded data indicator of the request header.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*NONE (default)</td>
<td>RU not padded</td>
</tr>
<tr>
<td>R*PADDED</td>
<td>RU was padded before encipherment</td>
</tr>
</tbody>
</table>

**Sense Data Transmit**

Sense data is inserted after the RH if the S.SENSE–DATA–IND variable contains a non-zero value.

S.SD–ID (-- address)
Contains the sense data category.

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>User sense data only</td>
</tr>
<tr>
<td>08</td>
<td>Request reject</td>
</tr>
<tr>
<td>10</td>
<td>Request error</td>
</tr>
<tr>
<td>20</td>
<td>State error</td>
</tr>
<tr>
<td>40</td>
<td>RH (request header) usage error</td>
</tr>
<tr>
<td>80</td>
<td>Path error</td>
</tr>
</tbody>
</table>

S.SD–CODE (-- address)
Contains the sense data modifier and sense code specific information fields.

**NOTE**
The upper byte of this variable is not used when creating the sense data field with the BUILD_RH command.
Example:
Set the sense data category to request reject, sense data modifier to session limit exceeded, and fill in sense data code bytes with zero's.

```
8 S.SD-ID ! ( Make sense data category to request reject )
0X00050000 S.SD-CODE ! ( Create sense data modifier and sense code fields )
```

<table>
<thead>
<tr>
<th>Ignored</th>
<th>Sense</th>
<th>Sense</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Code</td>
<td>Modifier</td>
</tr>
</tbody>
</table>

\[ NOTE \]
See Appendix A for defined sense data categories. Refer to IBM's System Network Architecture, Reference Summary, for more information.

---

**Request/Response Unit Transmit**

The following variable is used for building a request/response unit when SNA fields are constructed using the BUILD_RU command.

**S.RU-ID ( -- address )**

Contains the response/request unit code. Valid ITL constants are shown in Tables 13-4 through 13-7.
The SDLC Emulation is implemented as a state-driven protocol emulation. Point-to-point and multipoint configuration is supported as well as two way alternate or two way simultaneous transmission. The emulation can be run as a primary station which controls the data link and issues commands to up to thirty-two secondary stations controlled by a primary. When acting as a secondary, the emulation receives commands and issues responses.

The emulation has been set up to run as:
- an automatic simulation which operates in accordance with IBM’s SDLC procedure for modulo 8;
- a semi-automatic tester. The test manager is used to build and execute test scenarios to test responses and for generation of errors; and to implement SNA information within SDLC information frames;
- a manual tester. The test is controlled from the user’s keyboard;
- up to 32 simultaneous links.

### 12.1 Emulation State Machines

To ensure correct protocol operation, state machines have been implemented. Based on input events (i.e. received frames and timers) transitions from one state to another are made in accordance with IBM's SDLC implementation. Up to 32 state machines are supported simultaneously (i.e. one state machine per link). Separate NR and NS counts are maintained for each link.

Different states are applicable for the primary and secondary ends of the emulation. Emulation states are defined by ITL constants, as shown in the following tables.
<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_LINK</td>
<td>No link is established with the secondary. No I frames can be sent</td>
</tr>
<tr>
<td>LINK</td>
<td>A link has been established and polling is underway.</td>
</tr>
<tr>
<td>SEND_DISC</td>
<td>The primary is waiting for the secondary to relinquish control of the link so that is can send a DISC.</td>
</tr>
<tr>
<td>DISC/UA_WAIT</td>
<td>The primary has sent a DISC and is waiting for a UA. If a UA is not received, another DISC is sent out.</td>
</tr>
<tr>
<td>SIM/UA_WAIT</td>
<td>The primary has sent an SIM and it is waiting for the secondary to respond with a UA.</td>
</tr>
<tr>
<td>SEND_SNRM</td>
<td>The primary is waiting for control of the link so it can send out an SNRM frame.</td>
</tr>
<tr>
<td>SNRM/UA_WAIT</td>
<td>The primary has sent an SNRM and is waiting for the secondary to respond with a UA. If a UA is not received, another SNRM is sent out.</td>
</tr>
</tbody>
</table>

**Table 12-1 Primary Emulation States**

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDM</td>
<td>The normally disconnected mode state. The secondary responds with a DM or RIM when it is polled.</td>
</tr>
<tr>
<td>NRM</td>
<td>The normal response mode state and indicates that a link has been established. If polled an RR or RNR is returned. This is the only state of the secondary state machine in which I frames can be sent.</td>
</tr>
<tr>
<td>FRAME_REJECT</td>
<td>The frame reject state. If the secondary encounters an invalid frame from the primary, it enters this state and will only return FRMR’s. The emulation leaves this state when the primary sends an SIM, SNRM or DISC frame.</td>
</tr>
</tbody>
</table>

**Table 12-2 Secondary Emulation States**

Sometimes it is convenient to force the emulation into a specific state to observe the reaction of the DUT (device under test) under error conditions. The state machine can be forced into any state with the ENTER command.

**ENTER (state -- )**

Forces the automatic SDLC Emulation into a new state. Use the desired primary or secondary state from the previous tables as an input parameter.

⚠️ **WARNING**

The new state must belong to the same state machine set (i.e. primary or secondary) as unpredictable operation might result if a non-existent state is entered.

Example:

Force a primary SDLC Emulation into the NO_LINK state.

NO_LINK ENTER
12.2 Automatic Responses

The state machines normally handle the protocol automatically, i.e. automatic responses to received frames and timeouts.

The following commands activate and deactivate state machines.

EMUL_ON ( -- )
Activates the primary/secondary state machine (default). Automatic responses to received frames are generated.

EMUL_OFF ( -- )
Deactivates the primary/secondary state machine. No automatic responses to received frames are generated.

RUN_LAYER2 ( -- )
Forces processing of frames and timeouts through the emulation if the emulation has been turned off.

12.3 State-Dependent Send Commands

Either function keys or commands are used to transmit frames 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 protocol state machine must be activated (see Section 14 for examples).

FORM_LINK ( -- )
A primary emulation command; if issued on a secondary, the command is ignored. The action of this command depends upon whether or not a link has been established previously.

If a link has not been established previously (NO_LINK state), the primary station:
• sends out a broadcast XID (PU address hex FF); and
• expects to receive an XID from each secondary PU address defined on the Station Setup Menu (up to 32 supported).

On the currently selected station:
• when an XID is received, the primary responds with an RR and expects an RR in return; and
• when the RR is received, the primary sends out an SNRM and expects a UA (SNRM/UA-WAIT state). When the UA is received, polling starts (LINK state).

On all other supported stations:
• when an XID is received, the primary responds with a DISC and expects a DM in response. There is no change in state.
If a link has previously been established on any PU, the primary station:
- sends out an SNRM frame on the currently selected PU address and goes to
  the SNRM/UA_WAIT state; and
- once the expected UA is received, the emulation goes to the LINK state and polling
  (transmission of RR frames) starts on the currently selected PU address.

**NOTE**
The link should be formed separately for each PU address with which the primary station
intends to communicate.

Send topic
Enter PU function key
Form Link function key

**INIT_MODE (---)**
A secondary emulation command; if issued on a primary, the command is ignored. The result
of this command depends upon whether or not a link is up.

If a link has been established by the primary on the currently selected PU, the secondary
station using this PU (address):
- sends out an RIM and expects an SIM in response;
- responds with a UA after reception of the SIM;
- responds with a UA after the primary transmits an SNRM; and
- starts polling (transmission of RR frames between the primary and secondary).

If a link is not established by the primary on the currently selected PU, the secondary
emulation takes no action.

Send topic
Initialize Link function key

**KILL_LINK (---)**
This command can be issued on either a primary or secondary emulation. If a link is not
established on the currently selected PU, the command is ignored.

If any link has been established and this command is issued on a primary emulation, the
primary station:
- sends out a DISC frame on this PU and goes to the DISC/UA_WAIT state. Once this
  expected UA is received, polling on this PU ceases.

If a link has been established and this command is issued on a secondary emulation, the
secondary station:
- sends out an RD frame on the currently selected PU and expects to receive a DISC frame
  on this PU in response;
- responds to the DISC with a UA frame;
- responds with a DM frame if an RR frame is received on this PU;
- responds with a UA frame if another SNRM is received on this PU, the link is
  re-established, and polling continues.

Send topic
Kill Link function key
SNRM (--)  
A primary emulation command; if issued on a secondary, the command is ignored.

If a link has not been established previously, the primary station:
• remains in a NO_LINK state and no action is taken.

If a link has previously been established, the primary station:
• is forced into a NO_LINK state;
• resets the values of N(S) and N(R) to 0 for this PU and an SNRM is sent; and
• is then forced into the SNRM/UA_WAIT state and expects to receive a UA frame.

DISC (--)  
A primary emulation command; if used on a secondary, this command is ignored.

If a link has never been established, the emulation remains in a NO_LINK state and no action is taken.

If a link has been established and the primary has control of the link, the primary sends a DISC and enters the DISC/UA_WAIT state for this link.

If a link has been established and the secondary has control of the link, the primary is forced into the SEND_DISC state, i.e. it is waiting for the secondary to relinquish control of the link so that it can send a DISC.

SIM (--)  
A primary emulation command; if issued on a secondary, the command is ignored.

If a link has never been established, the primary station:
• sends a broadcast SIM with a PU address of hex FF and enters the SIM/UA_WAIT state;
• sends out an SNRM when a UA is received on the current PU, and goes to the SNRM/UA_WAIT state;
• sends a DISC frame when a UA is received on all other PU’s;
• sets the N(R) and N(S) values to 0 when the emulation receives a UA from the current PU and the emulation goes to the LINK state;
• sends an XID frame on the current PU; and
• starts polling on this PU when an XID frame is received on this PU.

If a link has been established, the primary station:
• is forced into the NO LINK state;
• sends out an SNRM on the current PU and goes to the SNRM/UA_WAIT state;
• transmits an SIM on the current PU and goes to the SIM/UA_WAIT state;
• returns to the SNRM_UA state when a UA is received and sends out an SNRM on the current PU; and
• goes to the LINK state after receiving a UA, and polling starts or continues on this PU.
XID ( -- )
A primary emulation command; if issued on a secondary, the command is ignored.

If a link has never been established via FORM_LINK; the command is ignored.

If a link has been established but is currently down:
• the emulation sends out an XID on the current address and expects an XID to be returned on the same PU.

If the emulation is in the LINK state, the primary station:
• sends out an XID on the current PU; and
• when it receives an XID on the same PU, polling resumes.

GET_ID ( -- )
A primary emulation command; if issued on a secondary, the command is ignored.

If a link has been established, an XID exchange is initiated.

RIM ( -- )
A secondary emulation command; if issued on a primary, the command is ignored.

If a link has never been established or if the currently selected link is down, the secondary remains in the NDM state and ignores the command.

If the link is up, the secondary station:
• is forced into NDM state;
• sends an RIM frame;
• sends a UA frame when an SIM frame is received from the primary; and
• responds with a UA frame if the secondary then receives an SNRM frame and goes to the NRM state and is ready to reply with an RR or RNR if polled.

RD ( -- )
A secondary emulation command; if issued on a primary, the command is ignored.

If a link has never been established or the currently selected link is down, the secondary remains in the NDM state and ignores the command.

If the link is up, the secondary station:
• transmits an RD at the first opportunity and waits for a DISC frame; and
• sends a UA frame when it receives a DISC frame from the primary and goes to the NDM state.

POLL_IN_1_FRAME ( -- )
Sends the last information frame in the window with the poll bit set to 1 (default).

NO_POLL_IN_1_FRAME ( -- )
Sends all information frames with the poll bit equal to 0. Following transmission of the last information frame in the window, an RR frame is sent with the poll bit set to 1, thus requesting acknowledgment.
OK (--)  
Once a link has been established, I frames can be accepted and the emulation sends out RR frames in response to a poll (default).

BUSY (--)  
Once a link has been established, no I frames can be accepted and the emulation sends out RNR frames in response to a poll.

RESET (--)  
If used with a primary emulation, resets all parameters and unconditionally terminates the link. Any unsent frames are cancelled.

If used with a secondary emulation:
- all parameters are reset;
- a DM frame is transmitted; and
- if a DISC frame is received, the link is terminated.

The following two commands transmit user-defined data. Up to 80 characters are allowed when directly input from the keyboard and 255 characters within test scripts.

SENDF (string --)  
Transmits a user-defined frame.

Example:
Transmit an SNRM with P bit set on PU address of hex BB.
X"BB93" SENDF

SEND_BTU (string --)  
Transmits a user-defined information field BTU (basic transmission unit) in an I frame. This I frame is sent over the current link with the correct N(R), N(S) and address of that link.

Example:
Send out an ACTLU request within all FID2 transmission header.
X"2D0002000B536B80000D0101" SEND_BTU

```
X"2D0002000B536B80000D0101"
FID2 TH Request ACTLU
```

IDACOM  
SDLC/SNA Programmer's Manual
The following four commands are used to set fields within an FID2. They must not be used to set these fields in other types of transmission headers. These commands are used in conjunction with SEND_BTU to transmit a frame containing the modified transmission header. The string must be created with dummy information in the respective fields.

**SET_DAF** (string\DAF value --string)
The string contains the correct number of bytes (6) for an FID2 plus the request/response header, request/response unit and sense data if desired. SET_DAF takes the specified DAF value and places it in the destination address field location of the specified string. Valid values for DAF within FID2 are 0 through 255.

Example:
Transmit a frame containing an FID2 transmission header with a predefined DAF, a response header and an ACTPU response unit.

X" 2D0000000B52EB800011020105000000000015" (Define string )SDAF @
(Obtain last defined SDAF ) SET_DAF (Modify string )SEND_BTU (Send frame )

```
DAF
  X" 2D0000000B52EB800011020105000000000015"
  FID2 TH Response Header ACTPU RU
```

**SET_OAF** (" string\"\OAF value --" string")
The string contains the correct number of bytes (6) for an FID2 plus the request/response header, request/response unit and sense data if desired. SET_OAF takes the specified OAF value and places it in the origin address field location of the specified string. Valid values for OAF are 0 through 255.

Example:
Transmit a frame containing an FID2 with a predefined OAF, a request header and an ACTLINK request unit.

X" 2D0002000B536B800001020A0102" (Define string )
SOAF @ (Obtain predefined OAF )
SET_OAF (Modify string )SEND_BTU (Send frame )

```
OAF
  X" 2D0002000B536B800001020A0102"
  FID2 TH Request Header ACTLINK R4
```
**SET_SNF** (string)\SNF value -- string)

The string contains the correct number of bytes for an FID2 plus the request/response header, request/response unit and sense data if required. SET_SNF takes the specified SNF value and places it in the sequence number field location of the specified string. Valid values for SNF are 0 through 65535.

Example:
Transmit a frame containing an FID2 with a predefined SNF; and a response header with sense data indicating a request reject category with the session limit exceeded.

\[ X"2D0000000B52EF900008050000" \]

1. Define string
2. Obtained predefined SNF
3. Modify string
4. Send frame

---

**RETURN_FIELDS** (string--string)

The string contains the correct number of bytes for an FID2 plus the request/response header, request/response unit, and sense data if desired.

This command:

- takes the last received DAF and places it in the origin address field location of the specified string;
- takes the last received OAF and places it in the destination address field location of the specified string; and
- takes the last received SNF and places it in the sequence number field of the specified string.

The modified string is left as an output parameter.
Example:
Transmit a frame with an FID2 transmission header with the OAF, DAF, and SNF of the last received frame echoed back and the received OAF and DAF swapped and used as the DAF and OAF of the transmitted frame.

\[ X'2D000000052EB800011020105000000000015' \] 
(Define string)

RETURN_FIELDS
(Swap and echo fields)

SEND_BTU
(Send frame)

The following four commands are used to construct SNA fields within an FID2 header. They can be used in conjunction with the SET_LENGTH command. To transmit a frame after the SNA fields have been constructed, the SEND_BTU command is used. See Section 11 for a description of variables and valid values used in construction.

**BUILD_TH-BYTE0** (" string "—" string" )
Forms byte 0 of the transmission header by modifying the first byte of the specified string.
- **S.FID-ID** — Only FID2 valid
- **S.MAPPING-FIELD** — Mapping field
- **S.EXP-FLOW-IND** — Expedited flow indicator

Example:
Build byte 0 of an FID2 with a mapping field indicating a first segment and normal flow.

\[ \begin{align*} 
0x20 & \text{ S.FID-ID} ! \\
R*FIRST-SEGMENT & \text{ S.MAPPING-FIELD} ! \\
R*NONE & \text{ S.EXP-FLOW-IND} ! \\
X'0000000000000000000000000000000000000000' & \text{ (Define string) } \\
\text{BUILD_TH-BYTE0} & \text{ (Build byte 0)} \\
\end{align*} \]

Transmit the frame:

**SET_LENGTH**
(Sets length of TH i.e. in this case 6 bytes)

**SEND_BTU**
(Send frame)

\[ \text{NOTE} \]
In the previous example, SET_LENGTH limits the length of the information field transmitted to six bytes, i.e. only the FID2.
**BUILD_TH** (string--string)

Builds an FID2 by modifying the first six bytes of the string reflecting the values in the following variables. The modified string is left as an output parameter.

- **S.FID-ID**: Only FID2 valid
- **S.MAPPING-FIELD**: Mapping field
- **S.EXP-FLOW-IND**: Expedited flow indicator
- **SOAF**: Origin address field
- **SDAF**: Destination address field
- **S.FORMAT-IND**: Format indicator from request/response header
- **SSNF**: Sequence number field
- **SSNF1**: Sequence number field for format 1, expedited flow

Example:

Build an FID2 with a mapping field indicating a whole segment, expedited flow, an origin address field of four, a destination address field of six, and format 1.

```
0x20 S.FID-ID ! FID2
R*WHOLE-BIU S.MAPPING-FIELD ! Whole segment
R*EXPEDITED S.EXP-FLOW-IND ! Expedited flow
4 SOAF !
6 SDAF !
R*FORMAT1 S.FORMAT-IND !
X"00000000000000000000000000000000000000000000000000000"
BUILD_TH
```

Transmit the frame.

- **SET_LENGTH** (Sets length of transmission header)
- **SEND_BTU** (Send in the frame)

**NOTE**

In this example, **SET_LENGTH** limits the length of the information field transmitted to 6 bytes, i.e. only the FID2.

**BUILD_RH** ("string"--"string")

Builds a request/response header by modifying the seventh to ninth bytes of the string reflecting the values in the following variables. The modified string is left as an output parameter.

- **S.RH-ID**: RU category
- **S.RH-REQ-RES**: Request or response indicator
- **S.FORMAT-IND**: Format indicator
- **S.SENSE-DATA-IND**: Sense data indicator
- **S.CHAIN-IND**: Chain indicator (begin and end)
- **S.RESPONSE-TYPE**: Response indicator
- **S.QUEUED-RESP-IND**: Queued response indicator
- **S.PACING-IND**: Pacing indicator
- **S.BRACKET-IND**: Bracket indicator (begin and end)
- **S.CHANGE-DIR-IND**: Change direction indicator
- **S.CODE-SEL-IND**: Code selection indicator
- **S.ENCIPH-DATA-IND**: Enciphered data indicator
- **S.PADDED-DATA-IND**: Padded data indicator
**Example:**

Build a session control request header.

```
0x60 S.RH-ID !  ( Session control )
0 S.RH-REQ-RES !  ( Request header )
R*FORMAT1 S_FORMAT-IND !  ( Format 1 )
R*NONE S.SENSE-DATA-IND !  ( No sense data )
R*ONLY-RU S.CHAIN-IND !  ( Only message in chain )
R*RQD1 S.RESPONSE-TYPE !  ( Definite response requested by DR1 )
R*NONE S.QUEUED-RESP-IND !  ( Transmission control queues bypassed )
R*NONE S.PACING-IND !  ( Pacing not requested )
R*NONE S.BRACKET-IND !  ( Between or within brackets )
R*NONE S.CHANGE-DIR-IND !  ( No direction change )
R*CODE0 S.CODE-SEL-IND !  ( Code 0 selected )
R*NONE S.ENCIPH-DATA-IND !  ( RU not enciphered )
R*NONE S.PADDDED-DATA-IND !  ( RU not padded )
X"2D00000000B52000000000000000000000000"  ( Define string )
BUILD_RH
```

Transmit the frame.

```
SET_LENGTH  ( Sets the length of transmission header and request header )
SEND_BTU  ( Send the frame )
```

⚠️ **NOTE**

In this example, SET_LENGTH limits the length of the information field transmitted to 9 bytes. If the S.SENSE-DATA-IND variable contained R*SENSE-DATA-INCL, the length would be 13 bytes to include 4 bytes of sense data.

**BUILD_RU** ( string—string )

Builds a request/response unit by modifying the appropriate location in the specified string using the following variables. The modified string is left as the output parameter.

- **S.RU-ID**  Response/request unit code
- **S.SENSE-DATA-IND**  Sense data indicator
Example:
Build a complete frame using BUILD_TH, BUILD_RH and BUILD_RU as described in previous examples.

R*FID2 S.FID-ID ! ( FID2 )
R*WHOLE-BIU S.MAPPING-FIELD ! ( Whole segment )
R*EXPEDITED S.EXP-FLOW-IND ! ( Expedited flow )
4 SOAP ! ( Origin address field )
6 SDAF ! ( Destination address field )
R*FORMAT1 S.FORMAT-IND ! ( Format 1 )
X" 000000000000000000000000000000000000000000000000000000000000000"
BUILD_TH ( Build transmission header )
R*SC S.RH-ID ! ( Session control )
R*REQ-HDR S.RH-REQ-RES ! ( Request header )
R*FORMAT1 S.FORMAT-IND ! ( Format 1 )
R*NONE S.SENSE-DATA-IND ! ( No sense data )
R*ONLY-RU S.CHAIN-IND ! ( Only message in chain )
R*RQD1 S.RESPONSE-TYPE ! ( Definite response requested by DR1 )
R*NONE S.QUEUED-RESP-IND ! ( Transmission control queues bypassed )
R*NONE S.PACING-IND ! ( Pacing not requested )
R*NONE S.BRACKET-IND ! ( Between or within brackets )
R*NONE S.CHANGE-DIR-IND ! ( No direction change )
R*CODE0 S.CODE-SEL-IND ! ( Code 0 Selected )
R*NONE S.ENCIPH-DATA-IND ! ( RU not enciphered )
R*NONE S.PADDED-DATA-IND ! ( RU not padded )
BUILD_RH ( Build request header )
R*ACTLU S.RU-ID ! ( ACTLU request code )
BUILD_RU ( Build response unit )

Transmit this frame.
SET_LENGTH ( Set length of frame )
SEND_BTU ( Send frame )

⚠️ NOTE
In this example, SET_LENGTH limits the length of the information field to 12 bytes, 6 for the FID2, 3 for the request header, and 3 for the request unit.

SET_LENGTH ( string--string )
This string is count prefixed, i.e. contains a count of the number of characters in the string. SET_LENGTH stores the length of the SNA fields (TH, RH, SD and RU) in this count.

⚠️ WARNING
Care must be used in using the SET_LENGTH command.
- If none of BUILD_TH-BYTE0, BUILD_TH, BUILD_RH, or BUILD_RU have been used since program startup, the length of the SNA fields is set to zero.
- If BUILD_TH-BYTE0 or BUILD_TH was the last SNA field commands to be called, the length of the SNA fields is set to 6.
- If BUILD_RH was the last SNA field command to be called, the length of the SNA fields is set to 9 if no sense data is included or 13 if sense data is included.
- If BUILD_RU was the last SNA field command to be called, the length of the SNA fields is set to include the FID2 transmission header, the request/response header, sense data if included and the request/response code.
NOTE
To change the length of the SNA fields after calling \texttt{SET\_LENGTH} or building the fields, use:
\texttt{N OVER C!}
This presumes the string is on the stack and has a length of at least \texttt{N} characters.

Example:
Build the SNA fields as shown under \texttt{BUILD\_RU}. Limit the length of the SNA fields to 10 bytes. The modified string created in the example remains on the stack for transmission with \texttt{SEND\_BTU}.

\begin{verbatim}
X" 2D0000000B520000000000000000000000 " ( Define string )
BUILD\_RH
SET\_LENGTH ( Sets the length of transmission header and request header )
10 OVER C! ( Set length of SNA fields to 10 )
SEND\_BTU ( Send the frame )
\end{verbatim}

### 12.4 State-Independent Send Commands

The following commands transmit frames without regard to correct protocol procedure. They do not update the protocol states or increment sequence values, i.e. N(R) AND N(S). They are designed to be used in test scenarios when frames are generated out of context. The variables used in these commands are described in Section 11.

\textbf{S:SNRM (--)}
Transmits an SNRM command frame using values from the following variables.
- **ID-OUT** – Address of the current link (valid values are 0 through 255)
- **P/F\_BIT** – P/F bit to be transmitted (0 or 1)

\textbf{S:UA (--)}
Transmits a UA response frame using values from the following variables.
- **ID-OUT** – Address of the current link (valid values are 0 through 255)
- **P/F\_BIT** – P/F bit to be transmitted (0 or 1)

\textbf{S:RD (--)}
Transmits an RD response frame using values from the following variables.
- **ID-OUT** – Address of the current link (valid values are 0 through 255)
- **P/F\_BIT** – P/F bit to be transmitted (0 or 1)

\textbf{S:DISC (--)}
Transmits a DISC command frame using values from the following variables.
- **ID-OUT** – Address of the current link (valid values are 0 through 255)
- **P/F\_BIT** – P/F bit to be transmitted (0 or 1)

\textbf{S:RIM (--)}
Transmits an RIM response frame using values from the following variables.
- **ID-OUT** – Address of the current link (valid values are 0 through 255)
- **P/F\_BIT** – P/F bit to be transmitted (0 or 1)
S:SIM ( -- )
Transmits an SIM command frame using values from the following variables.
ID-OUT – Address of the current link (valid values are 0 through 255)
P/F_BIT – P/F bit to be transmitted (0 or 1)

S:DM ( -- )
Transmits a DM response frame using values from the following variables.
ID-OUT – Address of the current link (valid values are 0 through 255)
P/F_BIT – P/F bit to be transmitted (0 or 1)

S:UP ( -- )
Transmits an UP command frame using values from the following variables.
ID-OUT – Address of the current link (valid values are 0 through 255)
P/F_BIT – P/F bit to be transmitted (0 or 1)

S:FRMR ( -- )
Transmits an FRMR response frame using values from the following variables.
ID-OUT – Address of the current link (valid values are 0 through 255)
P/F_BIT – P/F bit to be transmitted (0 or 1)
CONTROL – Value contained in the control field of the last received frame. See Tables A-1 and A-2 for defined values.
NR – Value of N(S) expected in the next received numbered I frame (valid values are 0 through 7).
NS – Value of N(S) in the next numbered I frame to be sent (valid values are 0 through 7).
BAD-CONTROL – Constant value of 1. When the frame is passed through the SNA Monitor, this is interpreted as an invalid frame control field.

S:CFGR ( -- )
Transmits a CFGR frame using values from the following variables.
ID-OUT – Address of the current link (valid values are 0 through 255)
P/F_BIT – P/F bit to be transmitted (0 or 1)

S:BCN ( -- )
Transmits a BCN response frame using values from the following variables.
ID-OUT – Address of the current link (valid values are 0 through 255)
P/F_BIT – P/F bit to be transmitted (0 or 1)

S:TEST ( -- )
Transmits a TEST frame with user-defined text using values from the following variables.
ID-OUT – Address of the current link (valid values are 0 through 255)
send_test – Buffer used for sending user-defined test message of up to 63 characters

Example:
Transmit a TEST frame containing the hex values of 01020304 in the test text field.

```
X" 01020304" ( Define TEST text )
COUNT 63 MIN DUP ( Obtain count up to 63 characters )
send_test C! ( Store count in first byte of buffer )
send_test 1+ SWAP CMOVE ( Store text in buffer starting at second byte )
S:TEST ( Send TEST frame )
```
S:UI (---)
Transmits a UI frame with a user-defined information field using values from the following variables.

- ID-OUT: Address of the current link (valid values are 0 through 255)
- P/F_BIT: P/F bit to be transmitted (0 or 1)
- send_ui: Buffer used for sending user-defined information field in UI frames (up to 63 characters).

Example:
Send a UI frame containing the hex value 01020304 in the information field and a P bit equal to zero.

```
0 P/F_BIT !           ( Set P bit to zero )
X" 01020304"          ( Define text for information field )
COUNT 63 MIN DUP      ( Obtain count of up to 63 characters )
send_ui C!             ( Store count in first byte of buffer )
send_ui 1+ SWAP CMOVE ( Store text in buffer starting at second byte )
S:UI
```

S:I (---)
Transmits a numbered I frame with a user-defined information field using values from the following variables.

- NR: Value of N(S) expected in next received numbered I frame (valid values are 0 through 7).
- NS: Value of N(S) in the I frame being sent (valid values are 0 through 7).
- ID-OUT: Address of the current link (valid values are 0 through 255)
- send_info: Buffer used for sending user-defined information field (up to 63 characters) in I frames.

Example:
Send an I frame containing the hex value 01020304 in the information field.

```
X" 01020304"          ( Define text for information field )
COUNT 63 MIN DUP      ( Obtain count of up to 63 characters )
send_info C!           ( Store count in first byte of buffer )
send_info 1+ SWAP CMOVE ( Store text in buffer )
S:I
```

S:RR (---)
Transmits an RR frame using values from the following variables.

- ID-OUT: Address of the current link (valid values are 0 through 255)
- P/F_BIT: P/F bit to be transmitted (0 or 1)
- NR: Value of N(S) expected in next received numbered I frame (valid values are 0 through 7).

S:RNR (---)
Transmits an RNR frame using values from the following variables.

- ID-OUT: Address of the current link (valid values are 0 through 255)
- P/F_BIT: P/F bit to be transmitted (0 or 1)
- NR: Value of N(S) expected in next received numbered I frame (valid values are 0 through 7).
S:REJ ( -- )
Transmits an REJ frame using values from the following variables.
ID-OUT   - Address of the current link (valid values are 0 through 255)
P/F.Bit   - P/F bit to be transmitted (0 or 1)
NR        - Value of N(S) expected in next received numbered I frame (valid values are 0 through 7).

12.5 CRC Errors

Frames can be sent with correct or incorrect CRC's (FCS), no CRC, or aborted during the transmission.

CRC-TYPE ( -- address )
Contains the error type status used by the current link when transmitting frames.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC (default)</td>
<td>A normal CRC is appended</td>
</tr>
<tr>
<td>CRC_ERR</td>
<td>A bad CRC is used</td>
</tr>
<tr>
<td>NO_CRC</td>
<td>No CRC is attached to the frame</td>
</tr>
<tr>
<td>ABT</td>
<td>The frame is aborted part way through</td>
</tr>
</tbody>
</table>

Example:
Force the current link to transmit CRC errors with each frame sent.
CRC_ERR CRC-TYPE !
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 SNA Monitor/SDLC Emulation applications.

For a complete explanation of the test manager and tools available, see the Programmer's Reference Manual.

This section review basic ITL components and describes the protocol event and action commands specific to the SNA Monitor and SDLC Emulation.

13.1 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( n -- )
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( n -- )
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( f -- )
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 STAT_INIT{ }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 ACTION{ }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 ACTION{ }ACTION definition and then called by the 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 ACTION{ }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 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.

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.

13.2 Event Recognition

During test script execution, any event received by the test manager is evaluated to determine if it matches the event-specified 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

If the SNA Monitor or SDLC Emulation is running on a B–Channel PRA test channel, no layer 1 events will be received by the test manager. See the Programmer's Reference Manual for a description of layer 1 events, i.e. control lead transitions when the application is running on a WAN interface.

Received Frames

ITL provides recognition of protocol specific frames, SNA protocol units, anchored or unanchored comparison of user–specified octets, CRC errors, and aborted frames.

Any frames received by the monitor or emulation are decoded and the decoded information is stored in various communication variables. The decoded information is used by the test manager to identify a particular event and can be obtained by the test program with the @ (fetch) command.

An identifier stored in the FRAME–ID variable is associated with each frame or BLU (basic link unit).
<table>
<thead>
<tr>
<th>Frame (BLU) Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*RR</td>
<td>Receive ready</td>
</tr>
<tr>
<td>R*RNR</td>
<td>Receive not ready</td>
</tr>
<tr>
<td>R*REJ</td>
<td>Reject</td>
</tr>
<tr>
<td>R*SNRM</td>
<td>Set normal response mode</td>
</tr>
<tr>
<td>R*DISC</td>
<td>Disconnect</td>
</tr>
<tr>
<td>R*SIM</td>
<td>Set initialization mode</td>
</tr>
<tr>
<td>R*UI</td>
<td>Unnumbered information</td>
</tr>
<tr>
<td>R*TEST</td>
<td>Test</td>
</tr>
<tr>
<td>R*XID</td>
<td>Exchange information</td>
</tr>
<tr>
<td>R*UA</td>
<td>Unnumbered acknowledgment</td>
</tr>
<tr>
<td>R*RD</td>
<td>Request disconnect</td>
</tr>
<tr>
<td>R*RIM</td>
<td>Request initialization mode</td>
</tr>
<tr>
<td>R*DM</td>
<td>Disconnect mode</td>
</tr>
<tr>
<td>R*UP</td>
<td>Unnumbered poll</td>
</tr>
<tr>
<td>R*FRMR</td>
<td>Frame reject</td>
</tr>
<tr>
<td>R*I</td>
<td>Numbered information present</td>
</tr>
<tr>
<td>R*SNRME</td>
<td>Set normal response mode eExtended</td>
</tr>
<tr>
<td>R*BCN</td>
<td>Beacon</td>
</tr>
<tr>
<td>R*CFGR</td>
<td>Configure</td>
</tr>
<tr>
<td>R*SHORTF RM</td>
<td>Short frame (no control field)</td>
</tr>
<tr>
<td>R*INVF RM</td>
<td>Invalid frame</td>
</tr>
<tr>
<td>R*INVCRC</td>
<td>Invalid CRC</td>
</tr>
</tbody>
</table>

Table 13-1 Frame (BLU) Identifiers

?RX_BLU (frame identifier -- flag)

Returns true if the specified frame is received. See Table 13-1 for a list of valid frame identifiers.

Example:
Check for reception of an RR frame.
R*RR ?RX_BLU

Example:
Check for reception of any supervisory frame (a logical 'OR' can be used).

1 STATE{
    R*RR ?RX_BLU
    R*RNR ?RX_BLU OR
    R*REJ ?RX_BLU OR
    ACTION{
        T. "Supervisory Frame received."
        TCR (Supervisory frame received? )
        ( Create trace statement )
        2 NEW_STATE
    }ACTION
}STATE
?RX_FRAME ( frame id#1\...\frame id#n\n -- flag )
Returns true if one of the specified frames is received. See Table 13-1 for a list of valid frame identifiers.

Example:
Check for reception of any supervisory frame.

1 STATE{
  R*RR R*RNR R*REJ 3 ?RX_FRAME  ( Supervisor frame received? )
  ACTION{
    T." Supervisory Frame received."  ( Create trace statement )
    TCR
    2 NEW_STATE
  }ACTION
}STATE

⚠ NOTE
This example has the same effect as the first example shown under ?RX_BLU.

?ADDRESS ( PU address -- flag )
Returns true if the specified PU address is the first byte of the received frame. Valid values are 0 through 255.

Example:
Check for reception of any frame with a PU address of 35.
35 ?ADDRESS

⚠ NOTE
This command can be used in conjunction with ?RX_BLU or ?RX_FRAME to screen out similar frames (BLU's).

Example:
Check whether an SNRM is received with the PU address of 200.
200 ?ADDRESS R*SNRM ?RX_BLU AND

Example:
Check whether an SNRM or DISC is received with the PU address of hex 50.
0x50 ?ADDRESS R*SNRM R*DISC 2 ?RX_FRAME AND

FRAME? ( -- flag )
Returns true if any frame is received.

DTE? ( -- flag )
Returns true if any frame is received from a DTE. This command can be used with a logical 'AND' in combination with other frame event recognition commands.

Example:
Check whether an SNRM is received with a PU address of 10 from a DTE.
R*SNRM ?RX_BLU
10 ?ADDRESS AND
DTE? AND
Example:
Check whether an SNRM or DISC is received with a PU address of 15 from a DTE.
R*SNRM R*DISC 2 ?RX_FRAME
15 ?ADDRESS AND
DTE? AND

DCE? (-- flag)
Returns true if any frame is received from a DCE. This command can be used with a logical 'AND' in combination with other frame event recognition commands.

Example:
Check whether an SNRM is received with a PU address of 10 from a DCE.
R*SNRM ?RX_BLU 10 ?ADDRESS AND DCE? AND

Example:
Check whether an SNRM or DISC is received with a PU address of 15 from a DCE.
R*SNRM R*DISC 2 ?RX_FRAME 15 ?ADDRESS AND DCE? AND

ABORT? (-- flag)
Returns true if an abort frame is received.

CRC_ERROR? (-- flag)
Returns true if a frame with a CRC error is received.

QEMPTY? (queue id -- flag)
Returns true if the specified queue is empty. Any I frames or UI frames to be transmitted by the emulation are queued and sent when possible. Once a UI frame is transmitted, it is removed from the queue and unavailable for retransmission. When an I frame is sent, it is maintained in a backup queue until it is acknowledged. Each data link maintains its own separate queue.

<table>
<thead>
<tr>
<th>Queue ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI_QUEUE</td>
<td>UI queue</td>
</tr>
<tr>
<td>I_QUEUE</td>
<td>I queue</td>
</tr>
<tr>
<td>I_BACKUP</td>
<td>I frame backup queue</td>
</tr>
</tbody>
</table>

Example:
Check to see whether an I frame can be transmitted (i.e. is the I queue empty).
I_QUEUE QEMPTY?
IF
   ( Send frame here )
ENDIF
Transmission Header Events

A format identifier stored in the FID–ID variable is associated with each TH (transmission header).

<table>
<thead>
<tr>
<th>Format Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*FID0</td>
<td>FID0 transmission header</td>
</tr>
<tr>
<td>R*FID1</td>
<td>FID1 transmission header</td>
</tr>
<tr>
<td>R*FID2</td>
<td>FID2 transmission header</td>
</tr>
<tr>
<td>R*FID3</td>
<td>FID3 transmission header</td>
</tr>
<tr>
<td>R*FID4</td>
<td>FID4 transmission header</td>
</tr>
<tr>
<td>R*FIDF</td>
<td>FIDF transmission header</td>
</tr>
<tr>
<td>R*INVFID</td>
<td>Invalid transmission header</td>
</tr>
</tbody>
</table>

Table 13–2 Transmission Header Identifiers

NOTE
If the received frame does not contain a transmission header, the FID–ID variable contains the value 0.

Example:
Check for reception of an FID4 transmission header.
FID–ID @ R*FID4 =

?THAF (address field -- flag )
Returns true if the specified destination or origin address field is found in the transmission header of a received frame. ?THAF is available in FID0, FID1, and FID2 transmission headers. Valid values are 0 through 65535 for an FID0 or FID1, and 0 through 255 for an FID2.

Example:
Check for reception of a frame with either a destination or origin address field of 100 in FID0, FID1, and FID2.
100 ?THAF

Example:
Check for reception of an FID2 transmission header with either a destination or origin address field of 100.
FID–ID @ R*FID2 =
100 ?THAF AND
?MAPPING_FIELD ( mapping field -- flag )
Returns true if the specified mapping field is found in the transmission header. The mapping field is a two bit field present in FID0, FID1, FID2, FID3, and FID4, but not FIDF.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*WHOLE-BIU</td>
<td>Entire message</td>
</tr>
<tr>
<td>R*FIRST-SEGMENT</td>
<td>First segment in message</td>
</tr>
<tr>
<td>R*LAST-SEGMENT</td>
<td>Last segment in message</td>
</tr>
<tr>
<td>R*MIDDLE-SEGMENT</td>
<td>Middle or intermediate segment in message</td>
</tr>
</tbody>
</table>

Example:
Check for reception of a transmission header containing a mapping field indicating the first segment in a message in FID0, FID1, FID2, FID3, and FID4.
R*FIRST-SEGMENT ?MAPPING_FIELD

Example:
Check for reception of an FID2 with a mapping field indicating the last segment in a message.
FID-ID @ FID2 =
R*LAST-SEGMENT ?MAPPING-FIELD AND

Request/Response Header Events

An identifier stored in the RH-ID variable is associated with each RH (request/response header).

<table>
<thead>
<tr>
<th>RU Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*FMD</td>
<td>Function management data, network services</td>
</tr>
<tr>
<td>R*NC</td>
<td>Network control</td>
</tr>
<tr>
<td>R*DFC</td>
<td>Data flow control</td>
</tr>
<tr>
<td>R*SC</td>
<td>Session control</td>
</tr>
<tr>
<td>R*INVRH</td>
<td>Invalid RU category</td>
</tr>
</tbody>
</table>

Table 13–3 Request/Response Unit Category Identifiers

?RH_REQ ( -- flag )
Returns true if the received frame contains a request header.

?RH_RES ( -- flag )
Returns true if the received frame contains a response header.
?RX_RH (RU category -- flag)
Returns true if the specified RU category is found in the received frame. See Table 13-3 for valid RU categories.

Example 1:
Check for reception of a frame with an RU category of FMD (function management data).
R*FMD ?RX_RH

Example 2:
Check for reception of a frame containing a response header and an RU category of FMD.
?RH_RES R*FMD ?RX_RH AND

Request/Response Unit Events
Protocol defined request/response units are associated with each category of RU.

?RX_RU (RU value -- flag)
Returns true if the specified request/response unit is found in the received frame. Valid RU values are listed in Table 13-4 to 13-7.

Example:
Check for reception of a frame containing an RU of abandon connection.
R*ABCONN ?RX_RU

The following table shows defined ITL constants for the function management data, network services RU category.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*ABCONN</td>
<td>Abandon connection</td>
</tr>
<tr>
<td>R*ABCONNOUt</td>
<td>Abandon connect out</td>
</tr>
<tr>
<td>R*ACTCONNIN</td>
<td>Activate connection in</td>
</tr>
<tr>
<td>R*ACTLINK</td>
<td>Activate link</td>
</tr>
<tr>
<td>R*ACTTRACE</td>
<td>Activate trace</td>
</tr>
<tr>
<td>R*ADDLINK</td>
<td>Add link</td>
</tr>
<tr>
<td>R*ADDLINKSTA</td>
<td>Add link station</td>
</tr>
<tr>
<td>R*ANA</td>
<td>Assign network addresses</td>
</tr>
<tr>
<td>R*BINDF</td>
<td>Bind failure</td>
</tr>
<tr>
<td>R*CDCINIT</td>
<td>Cross-domain control initiate</td>
</tr>
<tr>
<td>R*CDINIT</td>
<td>Cross-domain initiate</td>
</tr>
<tr>
<td>R*CDSESESEND</td>
<td>Cross-domain session end</td>
</tr>
<tr>
<td>R*CDSESSSF</td>
<td>Cross-domain session setup failure</td>
</tr>
<tr>
<td>R*CDSESSST</td>
<td>Cross-domain session started</td>
</tr>
<tr>
<td>R*CDSESTTF</td>
<td>Cross-domain session takedown failure</td>
</tr>
<tr>
<td>R*CDTAKED</td>
<td>Cross-domain takedown</td>
</tr>
<tr>
<td>R*CDTAKEDC</td>
<td>Cross-domain takedown complete</td>
</tr>
<tr>
<td>R*CDTERM</td>
<td>Cross-domain terminate</td>
</tr>
</tbody>
</table>

Table 13-4  FMD Request/Response Unit Identifiers
<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*CESLOW</td>
<td>Contents of FMD header = X&quot;01020C&quot;</td>
</tr>
<tr>
<td>R*CEXLOW</td>
<td>Contents of FMD header = X&quot;01020D&quot;</td>
</tr>
<tr>
<td>R*CINIT</td>
<td>Control initiate</td>
</tr>
<tr>
<td>R*CLEANUP</td>
<td>Clean up session</td>
</tr>
<tr>
<td>R*CONNOUT</td>
<td>Connect out</td>
</tr>
<tr>
<td>R*CONTACT</td>
<td>Contact</td>
</tr>
<tr>
<td>R*CONTACTED</td>
<td>Contacted</td>
</tr>
<tr>
<td>R*CTERM</td>
<td>Control terminate</td>
</tr>
<tr>
<td>R*DACTCONNIN</td>
<td>Deactivate connect in</td>
</tr>
<tr>
<td>R*DACTLINK</td>
<td>Deactivate link</td>
</tr>
<tr>
<td>R*DACTTRACE</td>
<td>Deactivate trace</td>
</tr>
<tr>
<td>R*DELETER</td>
<td>Delete network resource</td>
</tr>
<tr>
<td>R*DELIVER</td>
<td>Deliver</td>
</tr>
<tr>
<td>R*DISCONTACT</td>
<td>Discontact</td>
</tr>
<tr>
<td>R*DISPSTOR</td>
<td>Display storage</td>
</tr>
<tr>
<td>R*DSRLST</td>
<td>Direct search list</td>
</tr>
<tr>
<td>R*DUMPFINAL</td>
<td>Dump final</td>
</tr>
<tr>
<td>R*DUMPINIT</td>
<td>Dump initial</td>
</tr>
<tr>
<td>R*DUMPTEXT</td>
<td>Dump text</td>
</tr>
<tr>
<td>R*ECHOTEST</td>
<td>Echo test</td>
</tr>
<tr>
<td>R*ER-INOP</td>
<td>Explicit route inoperative</td>
</tr>
<tr>
<td>R*ER-TESTED</td>
<td>Explicit route tested</td>
</tr>
<tr>
<td>R*ESLOW</td>
<td>Entering slowdown</td>
</tr>
<tr>
<td>R*EXECTEST</td>
<td>Execute test</td>
</tr>
<tr>
<td>R*EXSLOW</td>
<td>Exiting slowdown</td>
</tr>
<tr>
<td>R*FNA</td>
<td>Free network address</td>
</tr>
<tr>
<td>R*FORWARD</td>
<td>Forward</td>
</tr>
<tr>
<td>R*INIT-OTHER</td>
<td>Initiate–other</td>
</tr>
<tr>
<td>R*INIT-OTHER-CD</td>
<td>Initiate–other cross–domain</td>
</tr>
<tr>
<td>R*INIT-SELF_F0</td>
<td>Initiate–self format 0</td>
</tr>
<tr>
<td>R*INIT-SELF_F1</td>
<td>Initiate–self format 1</td>
</tr>
<tr>
<td>R*INITPROC</td>
<td>Initiate procedure</td>
</tr>
<tr>
<td>R*INOP</td>
<td>Inoperative</td>
</tr>
<tr>
<td>R*IPLFINAL</td>
<td>IPL final</td>
</tr>
<tr>
<td>R*IPLINIT</td>
<td>IPL initial</td>
</tr>
<tr>
<td>R*IPLTEXT</td>
<td>IPL text</td>
</tr>
<tr>
<td>R*ISETCV</td>
<td>Contents of FMD header = X&quot;410222&quot;</td>
</tr>
<tr>
<td>R*LCP</td>
<td>Lost control point</td>
</tr>
<tr>
<td>R*LOREQD</td>
<td>Load required</td>
</tr>
<tr>
<td>R*NMVT</td>
<td>Network management vector transport</td>
</tr>
<tr>
<td>R*NOTIFY_SSCP-LU</td>
<td>Notify SSCP → LU</td>
</tr>
<tr>
<td>R*NOTIFY_SSCP-SSCP</td>
<td>Notify SSCP → SSCP</td>
</tr>
</tbody>
</table>

Table 13–4 FMD Request/Response Unit Identifiers [continued]
<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*NS-IPL-ABORT</td>
<td>NS IPL abort</td>
</tr>
<tr>
<td>R*NS-IPL-FINAL</td>
<td>NS IPL final</td>
</tr>
<tr>
<td>R*NS-IPL-INIT</td>
<td>NS IPL initial</td>
</tr>
<tr>
<td>R*NS-IPL-TEXT</td>
<td>NS IPL text</td>
</tr>
<tr>
<td>R*NS-LSA</td>
<td>NS lost subarea</td>
</tr>
<tr>
<td>R*NSLSA</td>
<td>Contents of FMD header = 410285</td>
</tr>
<tr>
<td>R*NSPE</td>
<td>NS procedure error</td>
</tr>
<tr>
<td>R*PROCSTAT</td>
<td>Procedure status</td>
</tr>
<tr>
<td>R*RECFMS</td>
<td>Record formatted maintenance statistics</td>
</tr>
<tr>
<td>R*RECMDD</td>
<td>Contents of FMD header = 010480</td>
</tr>
<tr>
<td>R*RECMMS</td>
<td>Record maintenance statistic</td>
</tr>
<tr>
<td>R*RECS</td>
<td>Record storage</td>
</tr>
<tr>
<td>R*RECFD</td>
<td>Record test data</td>
</tr>
<tr>
<td>R*RECTD</td>
<td>Record test results</td>
</tr>
<tr>
<td>R*RECTR</td>
<td>Record trace data</td>
</tr>
<tr>
<td>R*RECTRD</td>
<td>Record trace data</td>
</tr>
<tr>
<td>R*REQACTLU</td>
<td>Request activate logical unit</td>
</tr>
<tr>
<td>R*REQCONT</td>
<td>Request contact</td>
</tr>
<tr>
<td>R*REQDISCONT</td>
<td>Request disconnect</td>
</tr>
<tr>
<td>R*REQECHO</td>
<td>Request echo test</td>
</tr>
<tr>
<td>R*REQFNA</td>
<td>Request free network address</td>
</tr>
<tr>
<td>R*REQMS</td>
<td>Request maintenance statistics</td>
</tr>
<tr>
<td>R*REQTEST</td>
<td>Request test procedure</td>
</tr>
<tr>
<td>R*RNAA</td>
<td>Request network address assignment</td>
</tr>
<tr>
<td>R*ROUTE-TEST</td>
<td>Route test</td>
</tr>
<tr>
<td>R*RPO</td>
<td>Remote power off</td>
</tr>
<tr>
<td>R*SEESSEND</td>
<td>Session end</td>
</tr>
<tr>
<td>R*SESSST</td>
<td>Session started</td>
</tr>
<tr>
<td>R*SETCV-CONFIG</td>
<td>Set control vector (configuration)</td>
</tr>
<tr>
<td>R*SETCV-MAINT</td>
<td>Set control vector (maintenance)</td>
</tr>
<tr>
<td>R*STARTMEAS</td>
<td>Start measurement interval</td>
</tr>
<tr>
<td>R*STOPMEAS</td>
<td>Stop measurement interval</td>
</tr>
<tr>
<td>R*TERM-OTHER</td>
<td>Terminate other</td>
</tr>
<tr>
<td>R*TERM-OTHER-CD</td>
<td>Terminate other cross-domain</td>
</tr>
<tr>
<td>R*TERM-SELF</td>
<td>Terminate self format 0</td>
</tr>
<tr>
<td>R*TERM-SELF_F1</td>
<td>Terminate self format 1</td>
</tr>
<tr>
<td>R*TESTMODE</td>
<td>Test mode</td>
</tr>
<tr>
<td>R*UNBINDF</td>
<td>Unbind failure</td>
</tr>
<tr>
<td>R*VR-INOP</td>
<td>Virtual route inoperative</td>
</tr>
</tbody>
</table>

Table 13–4 FMD Request/Response Unit Identifiers [continued]
Table 13–5 shows defined RU (request/response unit) ITL constants for a DFC (data flow control) RU category.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*BID</td>
<td>Bid</td>
</tr>
<tr>
<td>R*BIS</td>
<td>Bracket initiation stopped</td>
</tr>
<tr>
<td>R*CANCEL</td>
<td>Cancel</td>
</tr>
<tr>
<td>R*CHASE</td>
<td>Chase</td>
</tr>
<tr>
<td>R*LUSTAT</td>
<td>Logical unit status</td>
</tr>
<tr>
<td>R*QC</td>
<td>Quiesce complete</td>
</tr>
<tr>
<td>R*QEC</td>
<td>Quiesce at end of chain</td>
</tr>
<tr>
<td>R*RELQ</td>
<td>Release quiesce</td>
</tr>
<tr>
<td>R*RSHUTD</td>
<td>Request shutdown</td>
</tr>
<tr>
<td>R*RTR</td>
<td>Ready to receive</td>
</tr>
<tr>
<td>R*SBI</td>
<td>Stop bracket initiation</td>
</tr>
<tr>
<td>R*SHUTC</td>
<td>Shutdown complete</td>
</tr>
<tr>
<td>R*SHUTD</td>
<td>Shutdown</td>
</tr>
<tr>
<td>R*SIG</td>
<td>Signal</td>
</tr>
</tbody>
</table>

Table 13–5 DFC Request/Response Unit Identifiers

Table 13–6 shows defined RU (request unit) ITL constants for the SC (session control) RU category.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*ACTCDRM</td>
<td>Activate cross-domain resource manager</td>
</tr>
<tr>
<td>R*ACTLU</td>
<td>Activate logical unit</td>
</tr>
<tr>
<td>R*ACTPU</td>
<td>Activate physical unit</td>
</tr>
<tr>
<td>R*BIND</td>
<td>Bind session</td>
</tr>
<tr>
<td>R*CLEAR</td>
<td>Clear</td>
</tr>
<tr>
<td>R*CRV</td>
<td>Cryptography verification</td>
</tr>
<tr>
<td>R*DACTCDRM</td>
<td>Deactivate cross-domain resource manager</td>
</tr>
<tr>
<td>R*DACTLU</td>
<td>Deactivate logical unit</td>
</tr>
<tr>
<td>R*DACTPU</td>
<td>Deactivate physical unit</td>
</tr>
<tr>
<td>R*RQR</td>
<td>Request recovery</td>
</tr>
<tr>
<td>R*SDT</td>
<td>Start data traffic</td>
</tr>
<tr>
<td>R*STSN</td>
<td>Set and test sequence numbers</td>
</tr>
<tr>
<td>R*UNBIND</td>
<td>Unbind session</td>
</tr>
</tbody>
</table>

Table 13–6 SC Request/Response Unit Identifiers
Table 13–7 shows defined RU (request unit) ITL constants for the NC (network control) RU category.

<table>
<thead>
<tr>
<th>ITL Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*ANSC</td>
<td>Contents of NC header = 06</td>
</tr>
<tr>
<td>R*LSA</td>
<td>Lost subarea</td>
</tr>
<tr>
<td>R*NC-ACTVR</td>
<td>NC activate virtual route</td>
</tr>
<tr>
<td>R*NC-DACTVR</td>
<td>NC deactivate virtual route</td>
</tr>
<tr>
<td>R*NC-ER-ACT</td>
<td>NC explicit route activate</td>
</tr>
<tr>
<td>R*NC-ER-ACT-REPLY</td>
<td>NC explicit route activate reply</td>
</tr>
<tr>
<td>R*NC-ER-INOP</td>
<td>NC explicit route inoperative</td>
</tr>
<tr>
<td>R*NC-ER-OP</td>
<td>NC explicit route operative</td>
</tr>
<tr>
<td>R*NC-ER-TEST</td>
<td>NC explicit route test</td>
</tr>
<tr>
<td>R*NC-ER-TEST-REPLY</td>
<td>NC explicit route test reply</td>
</tr>
<tr>
<td>R*NC-IPL-ABORT</td>
<td>NC IPL abort</td>
</tr>
<tr>
<td>R*NC-IPL-FINAL</td>
<td>NC IPL final</td>
</tr>
<tr>
<td>R*NC-IPL-INIT</td>
<td>NC IPL initial</td>
</tr>
<tr>
<td>R*NC-IPL-TEXT</td>
<td>NC IPL text</td>
</tr>
</tbody>
</table>

Table 13–7 NC Request/Response Unit Identifiers

?MATCH_I ( string= flag )
Returns true if a user-defined character string is found.

This is an anchored match, i.e. a byte-for-byte match starting at the first byte of the information field of the received I frame.

NOTE
To accommodate "don't care" character positions, the question mark character for ASCII, or hex 3F character can be used.

Example:
Match an ACTLU response unit within an FID2 transmission header.

\[ X^* \text{3F3F3F3F3FEB3F3FO}^* \text{FID2 RH} \ ?\text{MATCH}_I \]

The first six "don't care" characters include an FID2 transmission header. The hex 'EB' in byte 0 of the RH indicates that this frame contains a response header with a session control RU. The next two "don't care" characters include bytes 1 and 2 of the RH. The hex '0D' is the code for an ACTLU response unit.
Example:
This example shows an alternate method of matching an ACTLU response unit within an FID2. This alternate code is useful for it does not require knowledge of actual bits within a desired frame.

\[
\text{FID-ID @ R*FID2} = \quad ( \text{Is this an FID2? } ) \\
\text{?RH_RES \ AND} \quad ( \text{Is this also a Response? } ) \\
\text{R*SC \ ?RX_RH \ AND} \quad ( \text{Is this also a Session Control RU? } ) \\
\text{R*ACTLU \ ?RX_RU \ AND} \quad ( \text{Is this also an ACTLU? } )
\]

\(?\text{SEARCH}_I\) \ (\text{string}-- \text{flag})
Returns true if a user-defined character string is found within the information field of an I frame.

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

Example:
Search for an ACTLU RU within a frame.
\(x^* \text{ OD}^* \ ?\text{SEARCH}_I\)

This code returns a true flag if an ACTLU frame is received. However, it also returns a true flag for any I frame with the value hex 0D at any location within the information field.

---

Timeout Detection

There are 128 user programmable timers available. Timers 1 through 24 can be used in the test manager. Timer 32 is the test manager wakeup timer. The remaining timers are used in the application and should not be started or stopped in a test script.

\(?\text{TIMEOUT} \ (-- \text{flag})\)
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.

\(\text{8 \ STATE} \{
\quad \ ?\text{TIMEOUT} \quad ( \text{Check for timeout of any timer} ) \\
\quad \text{ACTION}{
\quad \quad \text{T.}^* \text{ A Timer has expired.}^* \ \text{TCR}
\quad \} \text{ACTION}
\} \text{STATE}
\)
**?TIMER (timer# -- flag)**

Returns true if the specified timer has expired. Valid input parameters are timers 1 through 24.

Example:
In State 8, look for the expiration of timer 21. The action is to display a trace statement.

```
8 STATE[
  21 ?TIMER  (Check for timeout of timer 21 )
  ACTION[
    T." Timer 21 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 34 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.

```
0 STATE[
  ?WAKEUP  (Check for timeout of wakeup timer )
  ACTION[
    T." To start the test, press Fl." TCR
    1 NEW_STATE
  ]ACTION
]STATE
```

**TIMER-NUMBER ( -- address)**

Contains the number of the expired timer. Valid values are 1 through 128.

---

**Function Key Detection**

Refer to the Programmer's Reference Manual.

---

**Interprocessor Mail Events**

Refer to the Programmer's Reference Manual.
**Wildcard Events**

The SNA Monitor and SDLC Emulation support the OTHER_EVENT command and the EVENT-TYPE variable. Refer to the Programmer's Reference Manual.

The EVENT-TYPE variable contains any one of the following constants: EOF_IND, TIMER_IND, I/F_IND, FK_IND or COMMAND_IND.

**EOF_IND (-- value)**
- A constant value in the EVENT-TYPE variable when the received event is a frame. The actual protocol type is in the FRAME-ID, FID-ID and RH-ID variables.

**TIMER_IND (-- value)**
- A constant value in the EVENT-TYPE variable when the received frame is a timeout.

**I/F_IND (-- value)**
- A constant value in the EVENT-TYPE variable when the received event is a control lead transition. The actual lead transition is in the LEAD-NUMBER variable.

**FK_IND (-- value)**
- A constant value in the EVENT-TYPE variable when a function or cursor key is detected.

⚠️ **NOTE**
*To detect function keys, it is advisable to use the ?KEY command. Refer to the Programmer's Reference Manual.*

**COMMAND_IND (-- value)**
- A constant value in the EVENT-TYPE variable when an interprocessor mail indication is received. Refer to the Programmer's Reference Manual.

### 13.3 SDLC/SNA Actions

All of the general actions explained in the Programmer's Reference Manual are supported in the SNA Monitor and SDLC Emulation. For additional display commands specific to SDLC/SNA, refer to Section 7 of this manual.
Layer 1 Actions

The following emulation commands turn control leads on and off.

<table>
<thead>
<tr>
<th>V.28/RS–232C Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OFF to ON</strong></td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>RTS_ON</td>
</tr>
<tr>
<td>CTS_ON</td>
</tr>
<tr>
<td>DSR_ON</td>
</tr>
<tr>
<td>CD_ON</td>
</tr>
<tr>
<td>DTR_ON</td>
</tr>
<tr>
<td>SQ_ON</td>
</tr>
<tr>
<td>RI_ON</td>
</tr>
<tr>
<td>DRS_ON</td>
</tr>
</tbody>
</table>

Table 13–8 V.28/RS–232C Interface Lead Transitions

<table>
<thead>
<tr>
<th>V.35 Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OFF to ON</strong></td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>RTS_ON</td>
</tr>
<tr>
<td>CTS_ON</td>
</tr>
<tr>
<td>DSR_ON</td>
</tr>
<tr>
<td>CD_ON</td>
</tr>
<tr>
<td>DTR_ON</td>
</tr>
<tr>
<td>RI_ON</td>
</tr>
</tbody>
</table>

Table 13–9 V.35 Interface Lead Transitions

<table>
<thead>
<tr>
<th>V.36/RS–449 Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OFF to ON</strong></td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>RTS_ON</td>
</tr>
<tr>
<td>CTS_ON</td>
</tr>
<tr>
<td>DSR_ON</td>
</tr>
<tr>
<td>DTR_ON</td>
</tr>
<tr>
<td>RI_ON</td>
</tr>
<tr>
<td>DRS_ON</td>
</tr>
<tr>
<td>CD_ON</td>
</tr>
</tbody>
</table>

Table 13–10 V.36/RS–449 Interface Lead Transitions
Protocol Actions

Frames can be transmitted using any of the commands described in Section 12.

NOTE
Any I frames or UI frames to be transmitted by the emulation are queued up and sent when possible. Once a UI frame is transmitted, it is removed from the queue and unavailable for retransmission. When an I frame is sent, it is maintained in a backup queue until it is acknowledged. Each data link maintains its own separate queue.

RESET_QUEUES (queue id --)
Resets the specified queue and discards any queued frames.

<table>
<thead>
<tr>
<th>Queue ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI_QUEUE</td>
<td>UI queue</td>
</tr>
<tr>
<td>I_QUEUE</td>
<td>I queue</td>
</tr>
<tr>
<td>L_BACKUP</td>
<td>I frame backup queue (the backup queue contains frames for retransmission)</td>
</tr>
</tbody>
</table>

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 to any size desired. However, the SDLC Emulation limits the number of bytes that can be transmitted to 4096.

A buffer consists of four bytes with values of 0, two bytes containing the length of the frame, and the remaining bytes consisting of user-defined contents.

Figure 13-1 Buffer Structure

NOTE
All buffers are cleared when the TCLR command is issued. TCLR is usually the first command executed 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.

Sending a Buffer
The text must first be stored in the buffer using STRING->BUFFER or FILE->BUFFER. Once the text is in place, the buffer can be transmitted repetitively.

The actual size of the frame sent is defined by the frame size set on the Emulation Configuration Menu or by the value stored in the MAX-FRAME-LENGTH variable.

BUFFER_SEND ( buffer number -- )
Transmits the specified buffer as an entire frame. Valid buffer numbers are 0 through 255.

Example:
Create the text to be included in the buffer first and then transmit the buffer.
X" 0100100100434445" 2 STRING->BUFFER
2 BUFFER_SEND

\textbf{NOTE}
When using BUFFER_SEND, the first byte of the buffer defines the frame address and the second byte the frame control.
SEND_BUFFER (buffer number --)
Transmits the specified buffer as the information field carried within an information frame that utilizes the current secondary address and N(R) and N(S) values.

Example:
Create the text in the buffer, then transmit the buffer.
```
" IDACOM" 4 STRING->BUFFER ( Create text )
4 SEND_BUFFER ( 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 SEND_BUFFER ( Send Data packet )
```
This section contains sample complete test scripts. These test scripts have also been supplied on disk and can be loaded and run as described in the Programmer's Reference Manual.

14.1 BAR_GRAPH.T

The BAR_GRAPH.T script illustrates the structure and usage of a test script in the test manager to collect statistics and display them in a bar chart fashion. Test script function keys are defined to display statistics or data.

TCLR

#IFDEF rr_count

0 VARIABLE rr_count
0 VARIABLE i_count
0 VARIABLE disc_count
0 VARIABLE snrm_count
0 VARIABLE ua_count
45 VARIABLE time
0 VARIABLE time_spot

#EndIF

0 SEQ{ ( -- )
0 rr_count !
0 i_count !
0 snrm_count !
0 disc_count !
0 ua_count !
10 time @ START_TIMER
2 time_spot @ THERE
  time @ 10 /MOD 2 W.R
   W." ." W.
}

SDLC/SNA Programmer's Manual
1 SEQ{ ( amount \ color -- )
WHERE DROP 9 THERE
OVER 4 W.R
" |" COUNT CYA_FG W.TYPE_A
OVER 50 >
IF
2DROP 50 RED_BG
ENDIF
OVER >R
" 
1+ ROT ROT W.TYPE_A
" 
1+ 50 R} - 0 W.TYPE_A
( Sequence to display a bar-graph )
( Position cursor for bar graph )
( Print out the amount )
( Is it too big for the graph? )
( Yes so limit it and make graph red )
( Save a copy of the length )
( Bar graph-52 spaces )
( Output the right number of dots )
( Unbar graph-52 spaces )
( Blank the rest of the bar graph )
}SEQ

0 STATE_INIT{
" SHOW DATA" 1 LABEL_KEY
" SHOW GRAPH" 2 LABEL_KEY
3 CLEAR_KEY 4 CLEAR_KEY 5 CLEAR_KEY
6 CLEAR_KEY 7 CLEAR_KEY 8 CLEAR_KEY
OPEN_USER CLEAR_TEXT 0 PAINT SHOW_USER ( Clear screen )
1 30 THERE ( Position for title )
" SDLC Line Statistics" COUNT WHI_FG W.TYPE_A
2 27 THERE ( Position for interval time )
" Frames per Interval Interval = " COUNT CYA_FG W.TYPE_A
WHERE 4 - time_spot ! DROP ( Remember time position )
" Seconds" COUNT CYA_FG W.TYPE_A
3 14 THERE
" 0___+___10___+___20___+___30___+___40___+___50"
COUNT CYA_FG W.TYPE_A ( Graph axis )
10 1 START_TIMER CLOSE_WINDOW
}STATE_INIT

0 STATE{
R*RR ?RX_BLU
ACTION{
  1 rr_count +!
}\ACTION

R*I ?RX_BLU
ACTION{
  1 i_count +!
}\ACTION

R*DISC ?RX_BLU
ACTION{
  1 disc_count +!
}\ACTION

IDACOM
R*SNRM ?RX_BLU
ACTION{
    1 snrm_count +!  (Increment the SNRM count)
}ACTION

R*UA ?RX_BLU
ACTION{
    1 ua_count +!  (Increment the UA count)
}ACTION

?TIMEOUT
ACTION
OPEN_USER
4 0 THERE W." RR's" rr_count @ GRN_BG 1 RUN_SEQ
5 0 THERE W." I FRAME's" i_count @ CYA_BG 1 RUN_SEQ
6 0 THERE W." DISC's" disc_count @ YEL_BG 1 RUN_SEQ
7 0 THERE W." SNRM's" snrm_count @ MAG_BG 1 RUN_SEQ
8 0 THERE W." UA's" ua_count @ WHI_BG 1 RUN_SEQ
0 RUN_SEQ CLOSE_WINDOW
}ACTION
}STATE
14.2 SNA_STAT.F

The SNA_STAT.F script illustrates the structure and usage of a test script in the SNA Monitor which collects and displays frame and transmission header statistics. This scenario can be used when monitoring line data or when playing back from disk recordings or capture RAM. Test script function keys are defined to display statistics, data or clear the statistic counts.

```
( SNA_STAT.F : SNA ANALYZER Test Manager Scenario )
( Collect and display SNA statistics )
( COUNTER1 to COUNTER30 collect counts. )
( f1 = Display Statistics. )
( f2 = Display Data. )
( f3 = Clear Statistic counts. )
```

```
#IFDEF ZERO_CNT
( Use conditional compiling to create counter )
( Variables and ZERO_CNT word in user dictionary )
( only if they have not been entered before. )
```

```
0 VARIABLE COUNTER11 0 VARIABLE COUNTER12 0 VARIABLE COUNTER13
0 VARIABLE COUNTER14 0 VARIABLE COUNTER15 0 VARIABLE COUNTER16
0 VARIABLE COUNTER17 0 VARIABLE COUNTER18 0 VARIABLE COUNTER19
0 VARIABLE COUNTER20 0 VARIABLE COUNTER21 0 VARIABLE COUNTER22
0 VARIABLE COUNTER23 0 VARIABLE COUNTER24 0 VARIABLE COUNTER25
0 VARIABLE COUNTER26 0 VARIABLE COUNTER27 0 VARIABLE COUNTER28
0 VARIABLE COUNTER29 0 VARIABLE COUNTER30
```

```
: ZERO_CNT ( -- )
( Zero statistic counts )
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 COUNTER20 !
0 COUNTER21 ! 0 COUNTER22 ! 0 COUNTER23 ! 0 COUNTER24 ! 0 COUNTER25 !
0 COUNTER26 ! 0 COUNTER27 ! 0 COUNTER28 ! 0 COUNTER29 ! 0 COUNTER30 !
```

```
#ENDIF
( End of the conditional compiling )
```

```
TCLR
( Clear test manager dictionary )
```

```
WAKEUP_ON
( Wakeup the test manager to display statistics on startup )
```

```
0 STATE_INIT{
```

```
    "STATISTICS" 1 LABEL_KEY
    "SHOW DATA"  2 LABEL_KEY
    "ZERO STAT"  3 LABEL_KEY
7 CLEAR_KEY  5 CLEAR_KEY  6 CLEAR_KEY  8 CLEAR_KEY
```

```
)STATE_INIT
```
3 SEQ

POP_USER
CLEAR_TEXT  WHI_FG  PAINT
1 24 THERE W."  SDLC/SNA Monitor Statistics"
3 10 THERE W."  SNRM = " COUNTER1 @ W. ( Display Statistics )
4 10 THERE W."  UA = " COUNTER2 @ W.
5 10 THERE W."  DISC = " COUNTER3 @ W.
6 10 THERE W."  RR = " COUNTER4 @ W.
7 10 THERE W."  RNR = " COUNTER5 @ W.
8 10 THERE W."  REJ = " COUNTER6 @ W.
9 10 THERE W."  DM = " COUNTER7 @ W.
10 10 THERE W."  FRMR = " COUNTER8 @ W.
11 10 THERE W."  SIM = " COUNTER9 @ W.
12 10 THERE W."  RIM = " COUNTER10 @ W.
13 10 THERE W."  UI = " COUNTER11 @ W.
14 10 THERE W."  UP = " COUNTER12 @ W.
15 10 THERE W."  RD = " COUNTER13 @ W.
16 10 THERE W."  XID = " COUNTER14 @ W.
17 10 THERE W."  TEST = " COUNTER15 @ W.
18 10 THERE W."  I FRAME = " COUNTER16 @ W.
19 10 THERE W."  BAD FRAME = " COUNTER17 @ W.
20 10 THERE W."  FRMR = " COUNTER18 @ W.
21 10 THERE W."  INVF = " COUNTER19 @ W.
22 10 THERE W."  DM = " COUNTER20 @ W.
23 10 THERE W."  DM = " COUNTER21 @ W.
24 10 THERE W."  DM = " COUNTER22 @ W.
25 10 THERE W."  DM = " COUNTER23 @ W.
26 10 THERE W."  DM = " COUNTER24 @ W.
CLOSE_WINDOW
)

4 SEQ

( Define sequence 4 to update and display I frame count )
1 COUNTER4 +! ( Statistic updating for I frames )
10 40 THERE COUNTER4 @ W. ( Display I frame count )
)

SEQ
5 SEQ{
    FID-ID @
    (Decode transmission header type)
    DOCASE
        CASE R*FIDO
            [ OPEN_USER 1 COUNTER18 +! 3 57 THERE COUNTER18 @ W. CLOSE_WINDOW ]
        CASE R*FID1
            [ OPEN_USER 1 COUNTER19 +! 4 57 THERE COUNTER19 @ W. CLOSE_WINDOW ]
        CASE R*FID2
            [ OPEN_USER 1 COUNTER20 +! 5 57 THERE COUNTER20 @ W. CLOSE_WINDOW ]
        CASE R*FID3
            [ OPEN_USER 1 COUNTER21 +! 6 57 THERE COUNTER21 @ W. CLOSE_WINDOW ]
        CASE R*FID4
            [ OPEN_USER 1 COUNTER22 +! 7 57 THERE COUNTER22 @ W. CLOSE_WINDOW ]
        CASE R*FIDF
            [ OPEN_USER 1 COUNTER23 +! 8 57 THERE COUNTER23 @ W. CLOSE_WINDOW ]
        CASE R*INVFD
            [ OPEN_USER 1 COUNTER24 +! 9 57 THERE COUNTER24 @ W. CLOSE_WINDOW ]
    ENDCASE
}SEQ

0 STATE{
    (Define state 0 to collect statistics and check events)
    R*SNRM ?RX_BLU
    (SNRM received)
    ACTION[
        OPEN_USER (Open user Display Window)
        1 COUNTER1 +! (Update SNRM counter)
        3 17 THERE COUNTER1 @ W. (Display SNRM count)
        CLOSE_WINDOW (Close Display Window)
    ]ACTION

    R*UA ?RX_BLU
    (UA received)
    ACTION[
        OPEN_USER (Open user Display Window)
        1 COUNTER2 +! (Update UA counter)
        4 17 THERE COUNTER2 @ W. (Display UA count)
        CLOSE_WINDOW (Close Display Window)
    ]ACTION

    R*DISC ?RX_BLU
    (DISC received)
    ACTION[
        OPEN_USER (Open user Display Window)
        1 COUNTER3 +! (Update DISC counter)
        5 17 THERE COUNTER3 @ W. (Display DISC count)
        CLOSE_WINDOW (Close Display Window)
    ]ACTION

    R*RR ?RX_BLU
    (RR frame received)
    ACTION[
        OPEN_USER (Open user Display Window)
        1 COUNTER5 +! (Update RR frame counter)
        6 17 THERE COUNTER5 @ W. (Display RR frame count)
        CLOSE_WINDOW (Close Display Window)
    ]ACTION
R*RNR?RX_BLU
ACTION{
    OPEN_USER
    1 COUNTER6 +!
    7 17 THERE COUNTER6 @ W.
    CLOSE_WINDOW
}ACTION

R*REJ ?RX_BLU
ACTION{
    OPEN_USER
    1 COUNTER7 +!
    8 17 THERE COUNTER7 @ W.
    CLOSE_WINDOW
}ACTION

R*DM ?RX_BLU
ACTION{
    OPEN_USER
    1 COUNTER8 +!
    9 17 THERE COUNTER8 @ W.
    CLOSE_WINDOW
}ACTION

R*FRMR ?RX_BLU
ACTION{
    OPEN_USER
    1 COUNTER9 +!
    10 17 THERE COUNTER9 @ W.
    CLOSE_WINDOW
}ACTION

R*SIM ?RX_BLU
ACTION{
    OPEN_USER
    1 COUNTER10 +!
    11 17 THERE COUNTER10 @ W.
    CLOSE_WINDOW
}ACTION

R*RIM ?RX_BLU
ACTION{
    OPEN_USER
    1 COUNTER11 +!
    3 42 THERE COUNTER11 @ W.
    CLOSE_WINDOW
}ACTION

RNR frame received
Open user Display Window
Display RNR frame count
Close Display Window

REJ frame received
Open user Display Window
Display REJ frame count
Close Display Window

DM frame received
Open user Display Window
Display DM frame count
Close Display Window

FRMR frame received
Open user Display Window
Display FRMR frame count
Close Display Window

Set Initialization Mode
Open user Display Window
Display SIM frame count
Close Display Window

Request Initialization Mode
Open user Display Window
Display RIM counter
Close Display Window
R*UI ?RX_BLU
ACTION{
  OPEN_USER
  1 COUNTER12 +!
  4 42 THERE COUNTER12 @ W.
  CLOSE_WINDOW
}ACTION

R*UP ?RX_BLU
ACTION{
  OPEN_USER
  1 COUNTER13 +!
  5 42 THERE COUNTER13 @ W.
  CLOSE_WINDOW
}ACTION

R*RD ?RX_BLU
ACTION{
  OPEN_USER
  1 COUNTER14 +!
  6 42 THERE COUNTER14 @ W.
  CLOSE_WINDOW
}ACTION

R*XID ?RX_BLU
ACTION{
  OPEN_USER
  1 COUNTER15 +!
  7 42 THERE COUNTER15 @ W.
  CLOSE_WINDOW
}ACTION

R*TEST ?RX_BLU
ACTION{
  OPEN_USER
  1 COUNTER16 +!
  8 42 THERE COUNTER16 @ W.
  CLOSE_WINDOW
}ACTION

R*I ?RX_BLU
ACTION{
  OPEN_USER
  1 COUNTER4 +!
  10 42 THERE COUNTER4 @ W.
  CLOSE_WINDOW
  5 RUN_SEQ
}ACTION

( Unnumbered Information Frame )
( Open user Display Window )
( Update UI counter )
( Display UI count )
( Close Display Window )

( Unnumbered Poll )
( Open user Display Window )
( Update UP counter )
( Display UP count )
( Close Display Window )

( Request Disconnect received )
( Open user Display Window )
( Update RD counter )
( Display RD count )
( Close Display Window )

( X I D frame )
( Open user Display Window )
( Update XID counter )
( Display XID count )
( Close Display Window )

( TEST Frame )
( Open user Display Window )
( Update Test frame count )
( Display Test frame count )
( Close Display Window )

( Information frame )
( Open user Display Window )
( Update I frame count )
( Display I frame count )
( Close display window )
( Decode transmission header )
R*INVFRM ?RX_BLU
ACTION{
  OPEN_USER
  1 COUNTER25 +!
  11 42 THERE COUNTER25 @ W.
  CLOSE_WINDOW
)
ACTION

UF3 ?KEY ?WAKEUP OR
ACTION{
  ZERO_CNT 3 RUN_SEQ
)
ACTION

UF2 ?KEY
ACTION{
  SHOW_DATA
)
ACTION

UF1 ?KEY
ACTION{
  3 RUN_SEQ
)
ACTION
)STATE

R*INVFRM ?RX_BLU
ACTION{
  OPEN_USER
  1 COUNTER25 +!
  11 42 THERE COUNTER25 @ W.
  CLOSE_WINDOW
)
ACTION

UF3 ?KEY ?WAKEUP OR
ACTION{
  ZERO_CNT 3 RUN_SEQ
)
ACTION

UF2 ?KEY
ACTION{
  SHOW_DATA
)
ACTION

UF1 ?KEY
ACTION{
  3 RUN_SEQ
)
ACTION
)STATE

( Invalid frame received )
( Open user Display Window )
( Update invalid frame count )
( Display invalid frame count )
( Close Display Window )

( CTRL f1 key or TM_RUN wakeup events )
( Open and show user Display Window )
( Show Data Window )
( Show user Display Window )
( END STATE 0 )
14.3 SNA 3274 Scenario

The file 3274.F can be used to test a 3274 terminal. The file 3274_DTE.F is provided to simulate a 3274 terminal.

This test script emulates a 37x5 communications controller generating traffic to a 327x and 3178 type CRT.

Pressing the START function key initiates the emulation and:

- a link is established;
- an ACTPU request is transmitted;
- when an ACTPU response is received, a 'Welcome' message is transmitted to the 3178;
- when either an INIT-SELF (format 0) request is received or a 'log-on' is entered on the 3178 keyboard, a BIND request is transmitted;
- when a BIND response is received, the function keys are relabelled. The user has the option of generating single or continuous screens of data which are transmitted to the 3178 CRT;
- entries typed on the emulation partner's keyboard receive a response which unlocks the keyboard; and
- when a 'logoff' message is typed on the 3178 keyboard, the tester initiates the UNBIND, DACTLU, DACTPU and DISC exchanges.

This test script has been designed to run on a switched SDLC line. To run on a non-switched SDLC line, replace the command '0 ID-OUT I' with one which specifies the station ID.

Example:
Set the link address to 41.

41 1 DO_SETID
File 3274.F

TCLR

BLU_BG TCOLOR ( Set trace statement color )
RESET ( Reset parameters and terminate link )
0 ID-OUT ! ( Set PU address to zero )
3 SDAF ! ( Set destination address )
1 SOAF ! ( Set origin address )
RFRM-INV RFRM-RR ( Block RR and invalid frames from display )
7 1 DO_SETW ( Set window size to 7 )

CTS_ON ( Turn on leads )
DSR_ON
CD_ON
SO_ON

" SEG_2" 0 FILE->BUFFER ( Create buffer )
" SEG_3" 1 FILE->BUFFER
" SEG_4" 2 FILE->BUFFER
" SEG_5" 3 FILE->BUFFER
" LAST_SET" 4 FILE->BUFFER
#IFNOTDEF S:ACTPU  ( Compile following code only if not in user dictionary )

: S:ACTPU ( -- )  ( Send Activate Physical Unit RU )
  X" 2D00000004D36B800011020105000000001"  ( Create string )
  SEND_BTU  ( Send frame )
/

: S:ACTLU ( -- )  ( Send Activate Logical Unit RU )
  X" 2D00000004D46B80000D02010"  ( Create string )
  SDAF @ SET_DAF  ( Set destination address )
  SEND_BTU  ( Send frame )
/

: S:INIT-SELF_F0_RES ( -- )  ( Send Initiate-Self Format 0 RU )
  X" 2C00000000008B80000010681"  ( Create string )
  RETURN_FIELDS  ( Echo back received DAF, OAF, and SNF )
  SEND_BTU  ( Send frame )
/

: S:BIND ( -- )  ( Send Bind Session RU )
  X" 2D00020104D56B800031010303B1903080000087F80000020000000000185
  000007E0000003E3E2D600"  ( Create string )
  SDAF @ SET_DAF  ( Set destination address )
  SOAF @ SET_OAF  ( Set origin address )
  SEND_BTU  ( Send frame )
/

: S:SDT ( -- )  ( Send Start Data Traffic RU )
  X" 2D00020104D66B80000A0"  ( Create string )
  SDAF @ SET_DAF  ( Set destination address )
  SOAF @ SET_OAF  ( Set origin address )
  SEND_BTU  ( Send frame )
/

: S:UNBIND ( -- )  ( Send Unbind Session RU )
  X" 2D00020104D76B80003202"  ( Create string )
  SDAF @ SET_DAF  ( Set destination address )
  SOAF @ SET_OAF  ( Set origin address )
  SEND_BTU  ( Send frame )
/

: S:DACTLU ( -- )  ( Send Deactivate Logical Unit RU )
  X" 2D00020004D86B800000E"  ( Create string )
  SDAF @ SET_DAF  ( Set destination address )
  SEND_BTU  ( Send frame )
/

: S:DACTPU ( -- )  ( Send Deactivate Physical Unit RU )
  X" 2D00000004D96B80001202"  ( Create string )
  SEND_BTU  ( Send frame )
/
:: FMD_ACK ( -- ) ( Send Response Header Acknowledgment )
  X' 2C0000000000838000" ( Create string )
  RETURN_FIELDS ( Echo back received DAF, OAF, and SNF )
  SEND_BTU ( Send frame )

:: DATA_RES ( -- ) ( Send data )
  X' 2C000301000A038040F5C111C2601DC8D9C5E2D7D6D5E2C540E3D640C5D5E3D9E8" ( Create string )
  SDAF @ SET_DAF ( Set destination address )
  SOAF @ SET_OAF ( Set origin address )
  SSNF @ SET_SNF ( Set sequence number )
  1 SSNF +! ( Increment sequence number )
  SEND_BTU ( Send frame )

:: LOGON_RES ( Send logon message )
  X' 2C000301000A038040F5C111C2601DC8D3D6C76D540C9E240E2E4C3C3C5E2C66E3D1D40111C3F013" ( Create string )
  SDAF @ SET_DAF ( Set destination address )
  SOAF @ SET_OAF ( Set origin address )
  SSNF @ SET_SNF ( Set sequence number )
  1 SSNF +! ( Increment sequence number )
  SEND_BTU ( Send frame )

:: LOGOFF_RES ( Send logoff message )
  X' 2C000301000A038040F5C111C2601DC8D3D6C76C640C9E240E2E4C3C3C5E2C66E3D1D40111C3F013" ( Create string )
  SDAF @ SET_DAF ( Set destination address )
  SOAF @ SET_OAF ( Set origin address )
  SSNF @ SET_SNF ( Set sequence number )
  1 SSNF +! ( Increment sequence number )
  SEND_BTU ( Send frame )
DATA AREA

SEND_SCREEN() {

"xxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
1 STATE{
  R*UA ?RX_BLU
  ACTION{
    S:ACTPU 2 NEW_STATE
  }ACTION
}STATE

2 STATE{
  ?RH_RES R*ACTPU ?RX_RU AND
  ACTION{
    S:ACTLU
    3 NEW_STATE
  }ACTION
}STATE

3 STATE{
  ?RH_RES R*ACTLU ?RX_RU AND
  ACTION{
    , fESCvTE@cV@chE@IDACVt@wC@srwT@DYIEEY@WYVGYAT
    ( Create welcome message )
    SDaF @ SET_DAF
    ( Set destination address )
    SEND_Bu
    ( Send WELCOME message )
    " WAITING FOR INIT-SELF OR 'logon' " W.NOTICE
    4 NEW_STATE
  }ACTION
}STATE

4 STATE{
  ?RH_REQ R*INIT-SELF_F0 ?RX_RU AND
  ACTION{
    S:INIT-SELF_F0_RES
    ( Send INIT-SELF response )
    S:BIND
    ( Send BIND request )
    5 NEW_STATE
  }ACTION

  R*FMD ?RX_RH ?RH_REQ AND " " ?SEARCH_I AND
  ACTION{
    S:BIND
    ( Send BIND request )
    1 COUNTER1!
    ( Status to indicate logon received )
    5 NEW_STATE
  }ACTION
}STATE

5 STATE{
  ?RH_RES R*BIND ?RX_RU AND
  ACTION{
    S:SDT
    ( Send SDT )
    6 NEW_STATE
  }ACTION
}STATE
6 STATE(
  ?RH_RES R*SDT ?RX RU AND
  ACTION(
    1 SSNF !
    COUNTER1 @
    IF
      S:LOGON_RES
    ENDF
    " Enter F2 to send Data once, F3 to send continuously" W.NOTICE
    1 CLEAR_KEY
    " ONE SCN"  2 LABEL_KEY
    " CONT SCN"  3 LABEL_KEY
    7 NEW_STATE
  )ACTION
)STATE

7 STATE(
  UF2 ?KEY
  ACTION(
    SEND_SCREEN
  )ACTION

  UF3 ?KEY
  ACTION(
    SEND_SCREEN
    " Send one screen of data"
    " DUP 2 LABEL_KEY 3 LABEL_KEY
    8 NEW_STATE
  )ACTION

  R*FMD ?RX RH ?RH_REQ AND " " ?SEARCH I AND
  ACTION(
    S:LOGOFF_RES
    S:UNBIND
    9 NEW_STATE
  )ACTION

  R*LAST-SEGMENT ?MAPPING-FIELD OR AND
  ( Does the RH contain an entire message or last segment of message? )

  ACTION(
    S:FMD ACK
    CHANGE-DIR-IND @
    IF
      S:DATA_RES
    ENDF
  )ACTION
)STATE
8 STATE{
  R*FMD ?RX_RH ?RH_RES AND SENSE-DATA-IND @ R*NONE = AND
      ( Is this an FMD response header without
       sense data? )
  ACTION{
    SEND SCREEN
  }ACTION

  R*FMD ?RX_RH ?RH_REQ AND " " ?SEARCH I AND
      ( Is this an FMD request header containing
       this string? )
  ACTION{
    S:LOGOFF_RES
    S:UNBIND
    9 NEW_STATE
    2 CLEAR_KEY 3 CLEAR_KEY
  }ACTION

  R*LAST-SEGMENT ?MAPPING-FIELD OR AND
      ( Does the RH contain an entire message or
       last segment of message? )
  ACTION{
    S:FMD_ACK
    CHANGE-DIR-IND @
    IF
      S:DATA_RES
    ENDIF
  }ACTION
}STATE

9 STATE{
  ?RH_RES R*UNBIND ?RX_RU AND
  ACTION{
    S:DACTLU
    10 NEW_STATE
  }ACTION
}STATE

10 STATE{
  ?RH_RES R*DACTLU ?RX_RU AND
  ACTION{
    S:DACTPU
    11 NEW_STATE
  }ACTION
}STATE
11 STATE(
  ?RH_RES R*DACTPU ?RX_RU AND
  ACTION[
    DISC
    12 NEW_STATE
  ]ACTION
)STATE

12 STATE(
  R*UA ?RX_BLU
  ACTION[
    T." Test is completed." TCR
    TM_STOP
  ]ACTION
)STATE

( Wait for a DACTPU response )
( Send DISC )
( Wait for UA )
( Reset parameters and terminate link )
( Stop test manager )
This test script is developed to simulate a 3274 terminal (for example, as a 327x cluster controller and 3178 type CRT emulation).

(FILE TITLE: 3274_DTE.F)
(DTE Emulation scenario used as test partner with 3274.F Scenario)

TCLR
BLU_BG TCOLOR

0xC1 1 DO_SETID  (1 Link address active)
1 1 DO_SET_LINKS  (Use C1 as the link address)
1 SDAF  (Set destination address)
3 SOAF  (Set origin address)

IFDEF S:LOGON

:LOGON
"2D00000000010380007DC6D311C5C89396879695"  (Define string)
SSNF @ SET_SNF  (Set sequence number)
SDAF @ SET_DAF  (Set destination address)
SOAF @ SET_OAF  (Set origin address)
SEND_BTU  (Send the frame)

:LOGOFF
"2D00000000010380007DC6D311C5C893968796866"  (Define string)
SSNF @ SET_SNF  (Set sequence number)
SDAF @ SET_DAF  (Set destination address)
SOAF @ SET_OAF  (Set origin address)
SEND_BTU  (Send frame)

:INIT_SELF
"2C0000003000106810004F4C3F2F7F8F2F303E3E2D6000009D5E2C1F761C1C4D7F6"  (Define string)
SDAF @ SET_DAF  (Set destination address)
SOAF @ SET_OAF  (Set origin address)
SEND_BTU  (Send frame)

ENDIF

0 STATE_INIT{  (Label function keys)

"LOGON" 1 LABEL_KEY
"LOGOFF" 2 LABEL_KEY
"INIT SELF" 3 LABEL_KEY
4 CLEAR_KEY 5 CLEAR_KEY 6 CLEAR_KEY
7 CLEAR_KEY 8 CLEAR_KEY
}

SDLC/SNA Programmer's Manual
0 STATE(
  UFl ?KEY
  ACTION[
    S:LOGON ( Send logon )
  ]ACTION

  UF2 ?KEY
  ACTION[
    S:LOGOFF ( Send logoff )
  ]ACTION

  UF3 ?KEY
  ACTION[
    S:INIT_SELF ( Send INIT_SELF )
  ]ACTION

R*ACTPU ?RX_RU ( ACTPU received? )
  ACTION[
    X"2D0000000B52EB800010201050000000015"
    RETURN_FIELDS
    SEND_BTU
  ]ACTION

R*ACTLU ?RX_RU ( ACTLU received? )
  ACTION[
    X"2D0002000B53EB80000D0201"
    RETURN_FIELDS
    SEND_BTU
  ]ACTION

R*BIND ?RX_RU ( BIND received? )
  ACTION[
    X"2D0008010001EBA0E031"
    RETURN_FIELDS
    SEND_BTU
  ]ACTION

R*SDT ?RX_RU ( SDT received? )
  ACTION[
    X"2D0008010001EBA060A0"
    RETURN_FIELDS
    SEND_BTU
  ]ACTION

R*UNBIND ?RX_RU ( UNBIND received? )
  ACTION[
    X"2D0008010001EBA0E032"
    RETURN_FIELDS
    SEND_BTU
  ]ACTION
```
R*DACTLU ?RX_RU
ACTION{
  X"2D0008010001EBA0E0E"
  RETURN_FIELDS
  SEND_BTU
}ACTION

R*DACTPU ?RX_RU
ACTION{
  X"2D0000000B52EB80001202"
  RETURN_FIELDS
  SEND_BTU
}ACTION

R*LAST-SEGMENT ?MAPPING-FIELD OR AND
( Request header containing entire message or last segment of message )
ACTION{
  X"2C0000000000838000"
  RETURN_FIELDS
  SEND_BTU
}ACTION
}STATE
```
14.4 Multipoint Test Scripts

The scenario files 'PRI_MULTI.F' and 'SEC_MULTI.F' are provided as sample test scripts to demonstrate the multipoint capabilities of the SDLC Emulation.

PRI_MULTI.F should be loaded and run on an IDACOM tester configured as a primary. SEC_MULTI.F should be loaded and run on an IDACOM tester configured as a secondary.

PRI_MULTI.F can also be run on the primary when no test script is running on the secondary. The user can select 'ONE QUEUE' or 'CONT QUES', F1 and F2 respectively. ONE QUEUE causes the primary to queue three information frames on each link which are transmitted as soon as there is an opportunity to do so. CONT QUES causes the primary to continuously queue information frames on all links.

If SEC_MULTI.F is running on the secondary, the secondary continuously queues information frames to be transmitted. Frames are then transmitted whenever there is an opportunity to do so.

The text of the information frames is taken from the 'fortune' program running on the UNIX™ operating system.

PRI_MULTI.F

( This script is designed to be run on the primary. It will use #SECS )
( for the number of links and set each link id to 1 + the link number. )
( The purpose of this script is to test the multipoint facilities of the )
( SDLC emulation. )

#IFDEF SET_STRINGS :
SET_STRINGS ( -- ) ( Fill nine buffers with data )
" 'The dA bore is someone who persists in holding his own views after we have en
lightened him with ours. ( 1 )" 0 STRING->BUFFER

" 'The dOne thing the inventors can't seem to get the bugs out of is fresh paint
( 2 )" 1 STRING->BUFFER

" 'The dRemember, even if you win the rat race -- you're still a rat. ( 3 )" 2 STRING->BUFFER

" 'The dThere are four kinds of homicide: felonious, excusable, justifiable, and
praiseworthy... -- Ambrose Pierce ( 4 )" 3 STRING->BUFFER

" 'The dYou couldn't even prove the White House staff sane beyond a reasonable d
oubt. -- Ed Meese, on the Hinckley verdict ( 5 )" 4 STRING->BUFFER
"The God made the world in six days, and was arrested on the seventh. (6)"
5 STRING->BUFFER

"The Truthful: Dumb and illiterate. (7)"
6 STRING->BUFFER

"The Kin: An affliction of the blood (8)"
7 STRING->BUFFER

"The Van Roy's Law: An unbreakable toy is useful for breaking other toys. (9)"
8 STRING->BUFFER

"The Good advice is something a man gives when he is too old to set a bad example. -- La Rouchefoucauld (A)"
9 STRING->BUFFER

#ENDIF

TCLR
YES RTRACE
YES DTRACE
YES CTRACE

0 SEQ{
  #SECS @ 0
  DO
    I_LINK# !
    I_BACKUP QEMPTY? I_QUEUE QEMPTY?
    AND
    IF
      "Primary queuing on link " LINK# @ T. TCR
      3 0
      DO
        J SEND_BUFFER
        LOOP
    ENDIF
  LOOP
}SEQ

1 SEQ{
  #SECS @ 0
  DO
    I_LINK# !
    RESET
  LOOP
}SEQ
0 STATE_INIT{
   ( Done on entry to state 0, sets initial conditions )
   CYA_FG TCOLOR
   MULTI_PT
   SET_STRINGS
   REP_COMP R=ASCII
   RH_OFF TH_OFF RU_OFF DATA_CHAR
   #SECS @ 10 >
   IF
      10 #SECS !
   ENDIF
   #SECS @ 0
   DO
      I I 1 + SET_LINK_ID
   LOOP
   " ONE QUEUE" 1 LABEL_KEY
   " CONT_QUES" 2 LABEL_KEY
   " STOP_ALL" 3 LABEL_KEY
   4 CLEAR_KEY 5 CLEAR_KEY 6 CLEAR_KEY 7 CLEAR_KEY 8 CLEAR_KEY
}STATE_INIT

0 STATE{
   UF1 ?KEY
   ACTION{
      0 RUN_SEQ
   }ACTION
   UF2 ?KEY
   ACTION{
      0 RUN_SEQ
      1 NEW_STATE
   }ACTION
   UF3 ?KEY
   ACTION{
      1 RUN_SEQ
   }ACTION
}STATE

1 STATE{
   UF1 ?KEY
   ACTION{
      0 NEW_STATE
   }ACTION
   UF3 ?KEY
   ACTION{
      1 RUN_SEQ
      0 NEW_STATE
   }ACTION
OTHER_EVENT
ACTION{
  0 RUN_SEQ
}ACTION
}STATE

( Queue more frames )
**SEC_MULTI.F**

( This script is designed to be run on the secondary. It will use #SECS )
( for the number of links and set each link id to 1 + the link number. )
( The purpose of this script is to test the multipoint facilities of the )
( SDLC emulation. )

#IFDEF SET_STRINGS

SET_STRINGS

: SET_STRINGS
  " 'The da bore is someone who persists in holding his own views after we have en
lightened him with ours. ( 1 )" 
0 STRING->BUFFER

  " 'The dOne thing the inventors can't seem to get the bugs out of is fresh paint
. ( 2 )" 
1 STRING->BUFFER

  " 'The dRemember, even if you win the rat race -- you're still a rat. ( 3 )" 
2 STRING->BUFFER

  " 'The dThere are four kinds of homicide: felonious, excusable, justifiable, and
praiseworthy... -- Ambrose Pierce ( 4 )" 
3 STRING->BUFFER

  " 'The dYou couldn't even prove the White House staff sane beyond a reasonable d
oubt. -- Ed Meese, on the Hinckley verdict ( 5 )" 
4 STRING->BUFFER

  " 'The dGod made the world in six days, and was arrested on the seventh. ( 6 )" 
5 STRING->BUFFER

  " 'The dTruthful: Dumb and illiterate. ( 7 )" 
6 STRING->BUFFER

  " 'The dKin: An affliction of the blood ( 8 )" 
7 STRING->BUFFER

  " 'The dVan Roy's Law: An unbreakable toy is useful for breaking other toys. ( 9
)" 
8 STRING->BUFFER

  " 'The dGood advice is something a man gives when he is too old to set a bad exa
mple. -- La Rouchefoucauld ( A )" 
9 STRING->BUFFER

; 

#ENDIF
TCLR
YES RTRACE  ( Report, record and capture all trace lines )
YES DTRACE
YES CTRACE

0 STATE_INIT[  ( Done on entry to state 0 )
  CYA_FG TCOLOR
  SET_STRINGS
  REP_COMP R=ASCII  ( Set up the monitor )
  RH_OFF RU_OFF TH_OFF DATA_CHAR  ( Turn off RH, RU, TH reports )
  #SECS @ 10 >
  IF
    10 #SECS !  ( Only buffer space for continuous queuing on 10 links )
  ENDIF
  #SECS @ 0
  DO
    I I 1 + SET_LINK_ID  ( Set the id's of all links )
  LOOP
}STATE_INIT

0 STATE[
  OTHER_EVENT
  ACTION{
    I_BACKUP QEMPTY?  I_QUEUE QEMPTY?
    AND  ( Are all the I-frames sent? )
    IF
      T." Secondary queuing frames on link " LINK# @ T. TCR
      3 0
    DO
      LINK# @ SEND_BUFFER  ( send 3 more I frames )
    LOOP
    ENDIF
  }ACTION
}STATE

SDLC/SNA Programmer's Manual

IDACOM
14.5 SDLC_SCRIPT.F

This test script illustrates transmission of information frames and the queuing structure of the SDLC Emulation.

Once the script is running, state 0 is entered and the initial conditions in STATE_INIT{ are performed. The link is put into a normal response mode and the monitor is set up to display the frame in character mode. Every time an event occurs, the scenario is run. The scenario continuously runs sequence 0 which checks to see if all the I frames have been sent (I_QUEUE QEMPTY?). If they have, then 5 more are sent.

TCLR

1 1 DO_SETID
   * THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS. 1234567890 the quick brown fox jumped over the lazy dogs. 1234567890 the quick brown fox jumped over the lazy dogs. 1234567890  Throughput message *
   0 STRING->BUFFER            ( Create a buffer )

   0 SEQ{
      I_QUEUE QEMPTY?           ( Are all the I-frames sent? )
      IF
        5 0
        DO
          0 SEND_BUFFER          ( Send 5 more )
          LOOP
        ENDIF
     )SEQ

   0 STATE_INIT{
      RESET SNRM               ( Set up the emulation )
      REP_CHAR R=ASCII         ( Set up the monitor )
     )STATE_INIT

   0 STATE{
      OTHER_EVENT
      ACTION{
        0 RUN_SEQ
      )ACTION
    )STATE
This appendix contains a brief introduction to IBM's implementation of SDLC and SNA. For further information, the reader is advised to consult IBM's publication GA27-3136-6, System Network Architecture, Reference Summary; and, publication GA27-3093-2, IBM Synchronous Data Link Control, General Information.

A data link consists of data communication equipment and the communication channel. The combination of DTE's, DCE's, and channel determine what is possible to accomplish with the data link. A number of communication configurations are possible depending on the capabilities of the connected devices.

A.1 SDLC Configurations

There are two types of configurations for SDLC:
- Point-to-point
- Multipoint

These configurations can operate in either two-way alternate (half duplex) or two-way simultaneous mode (full duplex). In two-way alternate mode, stations take turns transmitting. In two-way simultaneous mode, two stations can transmit and receive at the same time.
Point-to-Point

A point-to-point configuration is a data link with two stations. Links can be established on dedicated or dial-up circuits, and can be half duplex or full duplex.

Half duplex channels can be switched (temporary connection) or non-switched (permanent connection).

There are three basic point-to-point data link configurations:
- Half duplex, non-switched
- Full duplex, non-switched
- Half duplex, switched

Figure A-1 Point-to-Point Configuration
Multipoint

A multipoint link connects three or more stations.

There are two basic multipoint data link configurations:
- Half duplex, non-switched
- Full duplex, non-switched

![Multipoint Configuration](image)

Figure A-2 Multipoint Configuration

A.2 SNA Layers

SNA services are organized into a set of layers. IDACOM's implementation of the SNA Monitor is organized into the following layers:
- physical layer
- data link control layer
- path control layer
- transmission control layer
- data flow control layer
- function management layer
- application layer

SNA Fields
IDACOM's SDLC Emulation automatically supports the physical layer and data link control layer. SNA fields can be transmitted but must be user-defined.

**Physical Layer**

The physical layer provides physical, mechanical, and electrical conditions for the synchronous transmission of the bit streams generated by higher protocol layers. This layer is specified by reference to interface recommendations. IDACOM testers support V.28/RS-232C, V.11, V.36/RS-449, and V.35 interfaces.

IDACOM testers also support four methods of clocking: NRZ (standard non-return to zero) external transmit clock, NRZI (non-return to zero inverted), and NRZI with clock. Refer to 'Clocking' in Sections 2 and 9.2 for further information.
A.3 Data Link Control Layer

The data link control layer provides reliable communication between a primary and secondary station. The access procedure used is SDLC (synchronous data link control) and provides mechanisms to:

- initialize the link;
- ensure that any message can be transferred (transparency);
- minimize the probability of undetected errors;
- ensure that the information is transferred across the link in the correct sequence;
- control the flow of information across the link;
- detect and report procedure errors; and
- logically disconnect the link.

Frame Structure

All SDLC frames contain:

- a start flag;
- an address field;
- a control field;
- an information field (optional);
- a frame check sequence (FCS); and
- a closing flag.

![SDLC Frame Structure Diagram]

Figure A-4 SDLC Frame Structure
Flags

The flag prior to the address field is the opening flag, and the one following the FCS (frame check sequence) field is the closing flag. A single flag can be used as both the closing flag for one frame and the opening flag for the next frame. The bit pattern for these flags in SDLC is 0111 1110 (hex 7E).

When receiving, the last zero in the trailing flag can also be the first zero in the next leading flag as illustrated below.

```
|--leading flag--|
0111110111110
|--trailing flag--|
```

**NOTE**
Zero bits inserted by the transmission hardware prevent a flag pattern from occurring anywhere else in the frame.

Address

The second byte of the link header is the PU (physical unit) address field. For a primary, this address can be:

- a specific link station address – hex 01 to FE;
- a broadcast address – hex FF; or
- a 'no stations' address – hex 0.

Control Field

The control field identifies the type of frame and provides control information relevant to each type. The three types of frames are information, supervisory, and unnumbered.

Information frames:

- can be commands or responses.
- control byte contains:
  - an N(S) (send sequence number),
  - an N(R) (receive sequence number) – an acknowledgment of received frames,
  - a P/F (poll/final) bit used to request confirmation, and
- contain additional octets containing SNA information.
Supervisory frames:
- can be commands or responses;
- control field contains:
  - an N(R) (receive sequence number),
  - a P/F (poll/final) bit used to request confirmation, and
- include:
  - RR (receive ready),
  - RNR (receiver not ready),
  - REJ (reject).

Unnumbered frames:
- contain information used for link control, establishing or disconnecting a link;
- control field contains:
  - a P/F (poll/final) bit used to request confirmation,
- include:
  - UI (unnumbered information)
  - RIM (request initialization mode)
  - SIM (set initialization mode)
  - DM (disconnect mode)
  - UP (unnumbered poll)
  - RD (request disconnect)
  - DISC (disconnect)
  - UA (unnumbered acknowledgment)
  - SNRM (set normal response mode)
  - FRMR (frame reject)
  - XID (exchange identification)
  - CFGR (configure)
  - TEST (test)
  - BCN (beacon)
<table>
<thead>
<tr>
<th>Format</th>
<th>Command</th>
<th>Binary Configuration</th>
<th>Hex Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>Numbered Information Present</td>
<td>R R R P/F S S S 0</td>
<td>X'xx'</td>
</tr>
<tr>
<td>Transfer</td>
<td></td>
<td></td>
<td>X'xx'</td>
</tr>
<tr>
<td>Supervisory</td>
<td>RR (receive ready)</td>
<td>R R R P/F 0 0 0 1</td>
<td>X'x1'</td>
</tr>
<tr>
<td></td>
<td>RNR (receive not ready)</td>
<td>R R R P/F 0 1 0 1</td>
<td>X'x5'</td>
</tr>
<tr>
<td></td>
<td>REJ (reject)</td>
<td>R R R P/F 1 0 0 1</td>
<td>X'x9'</td>
</tr>
<tr>
<td>Unnumbered</td>
<td>UI (unnumbered information)</td>
<td>0 0 0 P/F 0 0 1 1</td>
<td>X'03'</td>
</tr>
<tr>
<td></td>
<td>RIM (request initialization mode)</td>
<td>0 0 0 F 0 1 1 1</td>
<td>X'07'</td>
</tr>
<tr>
<td></td>
<td>SIM (set initialization mode)</td>
<td>0 0 0 P 0 1 1 1</td>
<td>X'07'</td>
</tr>
<tr>
<td></td>
<td>DM (disconnect mode)</td>
<td>0 0 0 F 1 1 1 1</td>
<td>X'0F'</td>
</tr>
<tr>
<td></td>
<td>UP (unnumbered poll)</td>
<td>0 0 0 F 0 0 0 1</td>
<td>X'43'</td>
</tr>
<tr>
<td></td>
<td>RD (request disconnect)</td>
<td>0 1 0 F 0 0 1 1</td>
<td>X'43'</td>
</tr>
<tr>
<td></td>
<td>DISC (disconnect)</td>
<td>0 1 0 P 0 0 1 1</td>
<td>X'43'</td>
</tr>
<tr>
<td></td>
<td>SNRM (set normal response mode)</td>
<td>1 0 0 P 0 0 0 1</td>
<td>X'83'</td>
</tr>
<tr>
<td></td>
<td>FRMR (frame reject)</td>
<td>1 0 0 F 0 1 1 1</td>
<td>X'87'</td>
</tr>
<tr>
<td></td>
<td>XID (exchange identification)</td>
<td>1 0 1 P/F 1 1 1 1</td>
<td>X'AF'</td>
</tr>
<tr>
<td></td>
<td>CFRG (configure)</td>
<td>1 1 0 P/F 0 1 1 1</td>
<td>X'C7'</td>
</tr>
<tr>
<td></td>
<td>TEST (test)</td>
<td>1 1 1 P/F 0 0 1 1</td>
<td>X'EF'</td>
</tr>
<tr>
<td></td>
<td>BCN (beacon)</td>
<td>1 1 1 F 1 1 1 1</td>
<td>X'EF'</td>
</tr>
</tbody>
</table>

**NOTES:**

P = Poll bit  
F = Final bit  
RRR = Nr (receive count)  
SSS = Ns (send count)

Table A-1 SDLC Commands and Responses — Modulo 8
### INTRODUCTION TO IBM SNA

#### MSB LSB

<table>
<thead>
<tr>
<th>Bits</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Information</th>
<th>Command</th>
<th>Binary Configuration</th>
<th>Hex Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Transfer</td>
<td>Numbered Information Present</td>
<td>S S S S S S S 0 R R R R R R P/F</td>
<td>X'xxxx'</td>
</tr>
<tr>
<td>Supervisory</td>
<td>RR (receive ready)</td>
<td>0 0 0 0 0 0 0 1 R R R R R R P/F</td>
<td>X'01xx'</td>
</tr>
<tr>
<td></td>
<td>RNR (receive not ready)</td>
<td>0 0 0 0 0 1 0 1 R R R R R R P/F</td>
<td>X'05xx'</td>
</tr>
<tr>
<td></td>
<td>REJ (reject)</td>
<td>0 0 0 0 1 0 0 1 R R R R R R P/F</td>
<td>X'09xx'</td>
</tr>
</tbody>
</table>

**NOTES:**
P = Poll bit  
F = Final bit  
RRR = Nr (receive count)  
SSS = Ns (send count)

Table A–2 SDLC Commands and Responses – Modulo 128

In Tables A–1 and A–2, frames containing a P bit are commands and are issued by a primary. Frames containing an F bit are responses and are issued by a secondary. Frames containing P/F can be issued by either a primary or a secondary.

#### A.4 Path Control

Path control adds a TH (transmission header) to the BIU, creating a BTU (basic transmission unit).

SNA path control provides the following major services:
- routing data from node to node;
- segmenting and blocking messages;
- sequencing messages sent over multiple transmission links;
- controlling transmission priorities; and
- handling virtual route pacing.

SNA defines six types of TH (transmission header). These are FID0, FID1, FID2, FID3, FID4, and FIDF. The format and length of each type is determined by the FID (format identifier) which is located in byte 0 of the transmission header.

The FID type depends on the type(s) of nodes involved in the transmission:

**FID type 0** (length equals 10 bytes) is used for traffic involving non–SNA devices between adjacent subarea nodes when either or both nodes do not support explicit route and virtual route protocols.

**FID type 1** (length equals 10 bytes) is used for traffic between adjacent subarea nodes when either or both nodes do not support explicit route and virtual route protocols.

**FID type 2** (length equals 6 bytes) is used for traffic between a subarea node and an adjacent type 2.0 or 2.1 peripheral node.

**FID type 3** (length equals 2 bytes) is used for traffic between a subarea node and an adjacent type 1 peripheral node.
FID type 4 (length equals 26 bytes) is used for traffic between adjacent subarea nodes when both nodes support explicit route and virtual route protocols.

FID type F (length equals 26 bytes) is used for certain commands (for example, for transmission group control) sent between adjacent subarea nodes when both nodes support explicit route and virtual route protocols.

Figures A–5 to A–12 show the formats and descriptions of the bytes in each FID type.

---

Figure A–5 Transmission Header for FID Type 0

- **Format Identification (FID)**
  - 0000 = FID0

- **Expedited Flow Indicator (EFI)**
  - 0 = Normal Flow
  - 1 = Expedited Flow

- **Segmenting**
  - 00 = Middle
  - 01 = Last
  - 10 = First
  - 11 = Only
Figure A-6 Transmission Header for FID Type 1
### Figure A-7 Transmission Header for FID Type 2

- **Sequence Number Field (SNF)**
- **Origin Address Field (OAF)**
- **Destination Address Field (DAF)**
- **Reserved**
- **Flow Indicator (EFI)**
  - 0 = Normal Flow
  - 1 = Expedited Flow
  - **Segmenting**
    - 00 = Middle
    - 01 = Last
    - 10 = First
    - 11 = Only

Identifies the node that assigned the OAF-DAF pair associated with this session:
- 0 = Used by the assigning node having the greater role negotiation-value (carried in XID Format 3).
- 1 = Used by the other node.
**Figure A-8 Transmission Header for FID Type 3**

- **0000 0000** = SSCP–PU Session
- **01XX XXXX** = SSCP–LU Session
- **10XX XXXX** = Reserved
- **11XX XXXX** = LU–LU Session

- Expedited Flow Indicator (EFI)
  - 0 = Normal Flow
  - 1 = Expedited Flow

- Segmenting
  - 00 = Middle
  - 01 = Last
  - 10 = First
  - 11 = Only
Figure A-9 Transmission Header for FID Type 4 (Bytes 0 to 7)
Figure A-10  Transmission Header for FID Type 4 (Bytes 8–15)
Figure A-11 Transmission Header for FID Type 4 (Bytes 16 – 25)

Figure A-12 Transmission Header for FID Type F

* TG SNF Wrap Acknowledgment (only value defined)
A.5 Transmission Control

When a message and its associated parameters are sent down from data flow control, transmission control uses the information supplied in the parameters to construct an RH (request/response header).

The following major functions are performed by the transmission control:
- Sequencing
- Cryptography
- Session - level pacing

The RH (request/response header) is a 3 byte field immediately following the transmission header. In some cases, the transmission control layer creates a BIU (basic information unit) that contains only an RH and has no associated RU. These BIU's are used to perform control functions.

Path control can segment the BIU to form several PIU's (path information units) as shown in Figure A-13. When the BIU is segmented, the first segment contains the request/response header.

```
<table>
<thead>
<tr>
<th>TH</th>
<th>RH</th>
<th>RU Segment 1</th>
<th>TH</th>
<th>RU Segment 2</th>
<th>TH</th>
<th>RU Segment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PIU 1</td>
<td></td>
<td>PIU 2</td>
<td></td>
<td>PIU 3</td>
</tr>
</tbody>
</table>
```

Figure A-13 BIU Format – Multiple PIU's

If bit 0 in byte 0 of the RH equals zero, the RH is a request header and the associated RU is a request unit. If bit 0 of byte 0 of the RH equals 1, the RH is a response header and the associated RU is a response unit. Figure A-14 shows the formats and descriptions of the bytes in a request header. Figure A-15 shows the formats and descriptions of the bytes in a response header.
INTRODUCTION TO IBM SNA

September 1990

Pacing Indicator
1 = Pacing request

Queued Response Indicator
0 = Response bypasses transmission control queues
1 = Enqueue response in transmission control queues

Definite Response 1 Indicator
Definite Response 2 Indicator
Exception Response Indicator
0r00 = No response requested
0r10, 1r00, 1r10 = Definite response requested
Or11, 1r01, 1r11 = Exception response requested
(where r = reserved)

Enciphered Data Indicator
1 = RU is Enciphered

Padded Data Indicator
1 = RU was padded before encipherment

Conditional End Bracket Indicator
1 = Conditional End Bracket (for LU type 6.2, else reserved)

Change Direction Indicator
1 = Change Direction

Code Selection Indicator
0 = Code 0
1 = Code 1

Note A
00 = Middle
01 = Last
10 = First
11 = Only

RU in chain

Figure A-14 Request Header
Definite Response 1 Indicator (See Note A)
Definite Response 2 Indicator

Queued Response Indicator (QRI)
0 = Response bypasses TC queues
1 = Enqueue response in TC queues

Response Type Indicator (RTI)
0 = +RSP
1 = -RSP

Pacing Indicator (PI)
1 = Pacing response

Reserved

Byte 0 Byte 1 Byte 2

R1 Category Reserved (R1) Reserved (R0)

Note A
0r0 = Reserved except in IPR (with RTI = 0)
1r0 = DR1 response
0r1 = DR2 response
1r1 = DR1 and DR2 response (DR2 used with sync point)
where R = reserved

Only RU in chain

Sense Data Indicator
1 = Sense data included (+RSP only)

Format Indicator (Fl)
00 = FMD
01 = NC
10 = DFC
11 = SC

For LU–LU, (+RSP only), Fl = 0
For SSCP,
0 = character-coded RU
1 = Field-formatted RU
Fl = 1 (always)

Response

Figure A–15 Response Header
A.6 Data Flow Control

Data flow control communicates with both function management and transmission control by passing RU's (request/response units) and parameters. Data flow control functions include:
• processing a series of messages as a chain;
• processing a series of messages and responses as a bracket;
• determining and controlling the send/receive mode;
• assigning sequence numbers to RU's;
• interrupting data flow on request; and
• processing session status and error recovery.

A.7 Function Management

Function management constructs an RU (request/response unit) from the message created at the application layer. The RU is passed down through the data flow layer to transmission control.

SD (Sense Data)

If the sense data indicator in the RH (request/response header) is set to 1, 4 bytes of sense data are inserted after the RH and prior to the RU (request/response unit). The format of this field is shown in Figure A–16.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Modifier</td>
<td>Sense-code specific information</td>
<td>Sense Data</td>
</tr>
</tbody>
</table>

Figure A–16 Sense Data Format
Together, the category byte 0, the modifier byte 1, and the sense code specific bytes 2 and 3 hold the sense data defined for the exception condition that has occurred.

Refer to IBM's publication GA27-3136-B, System Network Architecture, Reference Summary, Chapter 9, Sense Data for more information on sense-code specific information.

---

**RU (Request/Response Unit)**

Request/response units follow the RH if sense data is not present. If sense data is present, the RU follows the sense data. For further information, refer to IBM's publication GA27-3136-B, System Network Architecture, Reference Summary, Chapter 4 – Requests and Chapter 5 – Responses.

---

**A.8 Application Layer**

A user operating at the application layer formulates a message and passes it to the function management layer of SNA.
### Table B-1 Clocking Modes

<table>
<thead>
<tr>
<th>Clocking Mode</th>
<th>Encoding Scheme</th>
<th>Clocking Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRZ with Clock</td>
<td>NRZ</td>
<td>DCE 15, 17</td>
</tr>
<tr>
<td>External Tx Clock</td>
<td>NRZ</td>
<td>DCE 15, 17</td>
</tr>
<tr>
<td>NRZI With Clock</td>
<td>NRZI</td>
<td>DCE 15, 17</td>
</tr>
<tr>
<td>NRZI</td>
<td>NRZI</td>
<td>DCE 15, 17</td>
</tr>
</tbody>
</table>

Clock speed is extracted from the data signal.

NRZ (Non-Return to Zero)  
- A 1-bit maps to a mark signal.  
- A 0-bit maps to a space signal.

NRZI (Non-Return to Zero Inverted)  
- A 1-bit maps to no transition.  
- A 0-bit maps to a transition.

15 – Transmit clock from DCE (DCE provided) CCITT circuit 114  
17 – Receive clock from DCE (DCE provided) CCITT circuit 114  
24 – Transmit clock to DCE (DTE provided) CCITT circuit 11

The pin numbers shown are for the RS-232C interface.

![Figure B-1 NRZ and NRZI Data Encoding](image_url)
SDLC/SNA TERMINOLOGY

A
Address byte of the SDLC link header.

ABCONN
Abandon connection.

ABCONNOT
Abandon connect out.

ACTCDRM
Activate cross-domain resource manager.

ACTCONNIN
Activate connection in.

ACTLINK
Activate link.

ACTLU
Activate logical unit.

ACTPU
Activate physical unit.

ACTTRACE
Activate trace.

ADD
Add link.

ADDLINK STA
Add link station.

ANA
Assign network addresses.

Basic Transmission Unit (BTU)
The unit of data and control information passed between path control components. A BTU can consist of one or more path information units (PIU's). See also path information unit.

BB
See begin bracket.

BBI
Begin bracket indicator.

BC
See begin chain.

BCI
Begin chain indicator.

Begin Bracket (BB)
The value (binary 1) of the begin bracket indicator in the request header (RH) of the first request in the first chain of a bracket; the value denotes the start of a bracket. Contrast with end bracket. See also bracket.

Begin Chain (BC)
The value (binary 1) of the begin chain indicator in the request header (RH) of the first of the RU chain; the value denotes the start of the chain. Contrast with end chain. See also RU chain.

BID
Bid

BIND
Bind session.

BINDF
Bind failure.

BIS
Bracket initiation stopped.

BIU
See basic information unit.
BIU Segment
The portion of a basic information unit (BIU) that is contained within a path information unit (PIU). It consists of either a request/response header (RH) followed by either all or part of a request/response unit (RU), or only part of an RU.

Bracket
A series of related normal-flow chains and their associated responses can be grouped together into a larger unit called a bracket. A bracket must be completed before another bracket can be started. Examples of brackets are data base inquiries/replies, update transactions, and remote job entry output sequences to work stations. See also begin bracket, end bracket, RU chain.

Bracket Protocol
A data flow control protocol in which exchanges between the two LU-LU half sessions are achieved through the use of brackets, with one LU designated at session activation as the first speaker and the other LU as the bidder. The bracket protocol involves bracket initiation and termination rules.

BTU
See basic transmission unit.

CANCEL
Cancel.

CDCINIT
Cross-domain control initiate.

CDI
Change direction indicator. See also change direction protocol.

CDINIT
Cross-domain initiate.

CDRM
Cross-domain resource manager.

CDSESENDD
Cross-domain session end.

CDSESSF
Cross-domain session setup failure.

CDSESSST
Cross-domain session started.

CDSESSF
Cross-domain session takedown failure.

CDTAKED
Cross-domain takedown.

CDTAKEDC
Cross-domain takedown complete.

CDTERM
Cross-domain terminate.

CEBI
Conditional end bracket indicator.

Chain
See RU chain.

Channel
See data channel.

Change Direction Protocol
A data flow control protocol in which the sending logical unit (LU) stops sending normal flow requests, signals this fact to the receiving LU using the change direction indicator (in the request header of the last request of the last chain), and prepares to receive requests.

CHASE
Chase.

CINIT
Control initiate.

CLEANUP
Clean up session.

CLEAR
Clear.

Cluster Controller Node
A peripheral node that can control a variety of devices.

Command
1. Any field set in the transmission header (TH), request header (RH), and sometimes portions of a request unit, that initiates an action or that begins a protocol; for example:
   a. bind session (session control request unit), a command that activates an LU-LU session,
   b. the change direction indicator in the RH of the last RU of a chain,
   c. the virtual route reset window indicator in an FID4 transmission header.
2. Loosely, a request unit.
3. In SDLC, the control information (in the C-field of the link header) sent from the primary station to the secondary station.
**SDLC/SNA TERMINOLOGY**

**Communication Adaptor**
An optional hardware feature, available on certain processors, that permits communication lines to be attached to processors.

**Communication Controller**
A type of communication control unit whose operations are controlled by one or more programs stored and executed in the unit; for example, the IBM 3705 Communications Controller.

**Configuration Services**
One of the types of network services in the system services control point (SSCP) and in the physical unit (PU); configuration services activate, deactivate, and maintain the status of physical units, links, and link stations. Configuration services also shut down and restart network elements and modify path control routing tables and address-transformation tables. See also maintenance services, management services, network services, physical unit control point, session services, SSCP.

**Communication Controller Node**
A term used to refer to a subarea node containing no system services control point (SSCP).

**CONNOU**
Connect out.

**CONTACT**
Contact.

**CONTACTED**
Contacted.

**Control Point**
A physical unit control point (PUCP) or a system services control point (SSCP).

**CRC**
Cyclic redundancy checking.

**Cross-Domain**
Pertaining to control or resources involving more than one domain.

**CRV**
Cryptography verification.

**Cryptographic Session**
An LU-LU session in which a function management data (FMD) request can be enciphered before it is transmitted and deciphered after it is received.

**Cryptography**
The transformation of data to conceal its meaning.

**CSI**
Code selection indicator.

**CTERM**
Control terminate.

**DACTCDRM**
Deactivate cross-domain resource manager.

**DACTCONNIN**
Deactivate connect in.

**DACTLINK**
Deactivate link.

**DACTLU**
Deactivate logical unit.

**DACTPU**
Deactivate physical unit.

**DACTTRACE**
Deactivate trace.

**DAF**
See destination address field.

**DAF'**
See destination address field prime.

**Data Channel**
A device that connects a processor and main storage with I/O control units.

**Data Circuit**
Synonym for link connection.

**Data Count Field (DCF)**
A binary count of the number of bytes in the basic information unit (BIU) or BIU segment associated with the transmission header (TH).

**Data Encrypting Key**
A key used to encipher and decipher data transmitted in a session that uses cryptography.
Data Flow Control (DFC)
A request/response unit (RU) category used for requests and responses exchanged between the data flow control layer in one half session and the data flow control layer in the session partner.

Data Flow Control (DFC) Layer
The layer within a half session that:
1. Controls whether the half session can send, receive, or concurrently send and receive request units (RUs).
2. Groups related RUs into RU chains.
3. Delimits transactions via the bracket protocol.
4. Controls the interlocking of requests and responses in accordance with control modes specified at session activation.
5. Generates sequence numbers.
6. Correlates requests and responses.

Data Link
Synonym for link.

Data Link Control (DLC) Layer
The layer that consists of the link stations that schedule data transfer over a link between two nodes and perform error control for the link. Examples of data link control are SDLC for serial–by–bit link connection and data link control for the System/370 channel.

Data Stream
A continuous stream of data elements being transmitted, or intended for transmission, in character or binary–digit form, using a defined format.

DFC
See data flow control.

Destination Address Field (DAF)
A field in an FID0 or FID1 transmission header that contains the network address of the destination. See also destination address field prime (DAF'), destination element field (DEF), destination subarea field (DSAF), format identification (FID) field, local session identification (LSID). Contrast with origin address field (OAF).

Destination Address Field Prime (DAF')
A field in an FID2 transmission header that contains the local address of the destination network addressable unit (NAU). See also destination address field (DAF), format identification (FID) field. Contrast with origin address field prime (OAF').

Destination Element Field (DEF)
A field in an FID4 transmission that contains an element address which, combined with the subarea address in the destination subarea field (DSAF), gives the complete network address of the destination network addressable unit (NAU). See also format identification (FID) field. Contrast with origin element field (OEF).

Destination Subarea Field (DSAF)
A field in an FID4 transmission header that contains a subarea address which, combined with the element address in the destination element field (DEF), gives the complete network address of the destination network addressable unit (NAU). See also format identification (FID) field. Contrast with origin subarea field (OSAF).

DCE
Data circuit terminating equipment.

DCF
See data count field.

DEF
See destination element field.

Definite Response
A value in the form–of–response–requested field of the request header. The value directs the receiver of the request to return a response unconditionally, whether positive or negative, to that request. Contrast with exception response, no response.

DESTENR
Delete network resource.
**DSAF**
See destination subarea field.

**DSRLST**
Direct search list.

**DTE**
Data terminal equipment.

**DUMPFINAL**
Dump final.

**DUMPINIT**
Dump initial.

**DUMPTEXT**
Dump text.

**EB**
See end bracket.

**EBCDIC**
Extended binary coded decimal interchange code.

**EBI**
End bracket indicator. See also end bracket.

**EC**
See end chain.

**ECHOTEST**
Echo test.

**ECI**
End chain indicator. See also end chain.

**ED**
The value (binary 1) of the enciphered data indicator in the request header (RH) denotes that the request unit (RU) is enciphered. See also cryptography.

**EDI**
Enciphered data indicator.

**EFI**
Expedited flow indicator. See also expedited flow.

**End Bracket**
The value (binary 1) of the end bracket indicator in the request header (RH) of the first request of the last chain of a bracket; the value denotes the end of the bracket. Contrast with begin bracket. See also bracket.

**End Chain**
The value (binary 1) of the end chain indicator in the request header (RH) of the last of the RU chain; the value indicates the end of the chain. Contrast with begin chain. See also RU chain.

**ER**
See explicit route.

**ER–INOP**
Explicit route inoperative.

**ER_TESTED**
Explicit route tested.

**ERI**
Exception response indicator. See also exception response.

**ERN**
Explicit route number.

**ESLOW**
Entering slowdown.

**Exception Request (EXR)**
A request that replaces another message unit in which an error has been detected.

**NOTE**
The exception request contains a 4 byte sense field that identifies the error in the original message unit and, except for some path errors, is sent to the destination of the original message unit; if possible, the sense data is returned in a negative response to the originator of the replaced message unit.

**Exception Response**
A value in the form of response requested field of a request header: the receiver is requested to return a response only if the request is unacceptable as received or cannot be processed; that is, a negative response, but not a positive response, could be returned.

**EXECTEST**
Execute test.
Expedited Flow
A data flow designated in the transmission header (TH) that is used to carry network control, session control, and various data flow control request/response units (RUs); the expedited flow is separate from the normal flow (which carries primarily end-user data) and can be used for commands that affect the normal flow. Contrast with normal flow.

NOTE
The normal and expedited flows move in both the primary to secondary and secondary to primary directions. Request and responses on a given flow (normal or expedited) usually are processed sequentially within the path, but the expedited flow traffic can be moved ahead of the normal flow traffic within the path at queuing points in the half sessions and for half session support in boundary functions.

Explicit Route (ER)
The path control network components, including a specific set of one or more transmission groups, that connect two subarea nodes. An explicit route is identified by an origin subarea address, a destination subarea address, an explicit route number, and a reverse explicit route number. See also path, virtual route.

Explicit Route Length
The number of transmission groups in an explicit route.

EXR
See exception request.

EXSLOW
Exiting slowdown.

F
See flag.

FCS
SDLC frame check sequence.

FID
Format identification. See format identification field.

FIFO
First in, first out.

Flag
Sync character of hexadecimal 7E.
Function Management Header
One or more headers, optionally present in the leading request units (RUs) of an RU chain, that allow one half session in an LU–LU session to:

1. Select a destination as the session partner and control the way the end user data it sends is handled at the destination.
2. Change the destination or the characteristics of the data during the session.
3. Transmit between session partners status or user information about the destination (for example, whether it is a program or a device).

Half Session
A component that provides FMD services, data flow control, and transmission control for one of the sessions of a network addressable unit (NAU).

HDX
Half duplex data flow.

I
SDLC information frame.

ID
Identification

IERN
Initial explicit route number.

INIT–OTHER
Initiate–other.

INIT–OTHER–CD
Initiate–other cross–domain.

INIT–SELF
Initiate–self.

Initiation
See LU–LU session initiation. See also session initiation request.

INITPROC
Initiate procedure.

INOP
Inoperative.

IPLFINAL
IPL final.

IPLINIT
IPL initial.

IPLTEXT
IPL text.

Layer
A grouping of related functions that are logically separate from the functions in other layers; the implementation of the functions in one layer can be changed without affecting functions in other layers. See also NAU services manager layer, FMD services layer, data flow control layer, transmission control layer, path control layer, data link control layer.

LCP
Lost control point.

LDREQD
Load required.

LH
See link header.

LIFO
Last in, first out.

Link
The combination of the link connection and the link stations joining network nodes; for example:

1. A System/370 channel and its associated protocols.

**NOTE**
A link connection is the physical medium of transmission; for example, a telephone wire or a microwave beam. A link includes the physical medium of transmission, the protocol, and associated communication devices and programming; it is both logical and physical.

Link Connection
The physical equipment providing two–way communication between one link station and one or more other link stations; for example, a communication line and data circuit terminated equipment (DCE). Synonymous with data circuit.

Link Header
Control information for data link control at the beginning of a basic link unit (BLU).

Link Station
The combination of hardware and software that allows a node to attach to and provide control for a link.

Link Trailer
Control information for data link control at the end of a basic link unit (BLU).
**Local Address**
An address used in a peripheral node in place of a network address and transformed to or from a network address by the boundary function in a subarea node. See also network address.

**Local Session Identification**
A field in an FID3 transmission header that contains an indication of the type of session (SSCP-PU, SSCP-LU, or LU-LU) and the local address of the peripheral logical unit (LU) or physical unit (PU).

**Logical Unit (LU)**
A port through which an end user accesses the SNA network in order to communicate with another end user and through which the end user accesses the functions provided by system services control points (SSCPs). An LU can support at least two sessions – one with an SSCP, and one with another logical unit and can be capable of supporting many sessions with other logical units.

**LSA**
Lost subarea.

**LSID**
See local session identification.

**LT**
See link trailer.

**LU**
See logical unit.

**LUSTAT**
Logical unit status.

**Maintenance Services**
One of the types of network services in system services control points (SSCP's) and physical unit (PU's). Maintenance services provide facilities for testing links and nodes and for collecting and recording error information. See also configuration services, management services, network services, session services.

**Management Services**
One of the types of network services in system services control points (SSCPs) and logical units (LUs). Management services forward requests for network data, such as error statistics, and deliver the data in reply. See also configuration services, maintenance services, network services, and session services.

**Message Segment**
That portion of a message that is contained within a single request unit.

**Message Unit**
A generic term for the unit of data processed by any layer; for example, a basic information unit (BIU), a path information unit (PIU), a request/response unit (RU).

**NA**
See network address.

**NAU**
See network addressable unit.

**NAU Services**
The functions provided by the NAU services manager layer and the FMD services layer.

**NAU Services Manager Layer**
The layer that:
1. Controls network operations via LU-LU, SSCP-LU, SSCP-PU, and SSCP-SSCP sessions.
2. Coordinates end user interactions on LU-LU sessions. See also configuration services, session services, maintenance services, management services.

**NC**
See network control.

**NC-ACTVR**
NC activate virtual route.

**NC-DACTVR**
NC deactivate virtual route.

**NC-ER-ACT**
NC explicit route activate.

**NC-ER-ACT-REPLY**
NC explicit route activate reply.

**NC-ER-INOP**
NC explicit route inoperative

**NC-ER-OP**
NC explicit route operative

**NC-ER-TEST**
NC explicit route test.

**NC-ER-TEST-REPLY**
NC explicit route test reply.

**NC-IPL-ABORT**
NC IPL abort.

**NC-IPL-FINAL**
NC IPL final.
NC-IPL-INIT
NC IPL initial.

NC-IPL-TEXT
NC IPL text.

Negative Response
A response indicating that a request did not arrive successfully or was not processed successfully by the receiver. Contrast with positive response. See also exception response.

Negotiable BIND
A capability that allows two LU-LU half sessions to negotiate the parameters of a session when the session is being activated.

Network Address
An address, consisting of subarea and element fields, that identifies a link, a link station, or a network addressable unit. Subarea nodes use network addresses; peripheral nodes use local addresses. The boundary function in the subarea node to which a peripheral node is attached transforms local addresses to network addresses and vice versa. See also local address.

Network Addressable Unit
A logical unit, a physical unit, or a system services control point. It is the origin or the destination of information transmitted by the path control network.

Network Control (NC)
A request/response unit (RU) category used for requests and responses exchanged between physical units (PUs) for such purposes as activating and deactivating explicit and virtual routes and sending load modules to adjacent peripheral nodes. See also data flow control, function management data, session control.

Network Services
The services within network addressable units (NAUs) that control network operation via SSCP-SSCP, SSCP-PU, and SSCP-LU sessions. See also configuration services, maintenance services, management services, session services.

Network Services Header
A 3 byte field in an FMD request/response unit (RU) flowing in an SSCP-LU, SSCP-PU, or SSCP-SSCP session. The network services header is used primarily to identify the network services category of the RU (for example, configuration services, session services) and the particular request code within a category.

NMVT
Network management vector transport.

Node
An endpoint of a link or a junction common to two or more links in a network. Nodes can be distributed to host processors, communication controllers, or terminals. Nodes can vary in routing and other functional capabilities.

No Response
A value in the form of response requested field of the request header (RH) indicating that no response is to be returned to the request, whether or not the request is received and processed successfully. Contrast with definite response, exception response.

Normal Flow
A data flow designated in the transmission header (TH) that is used primarily to carry end user data. The rate at which requests flow on the normal flow can be regulated by session level pacing. Contrast with expedited flow.

NOTE
Each NAU has a network address that represents it to the path control network (LUs can have multiple addresses for parallel LU-LU sessions). The path control network and the NAUs together constitute the SNA network.

NOTE
The normal and expedited flows move in both the primary to secondary and secondary to primary directions. Requests and responses on a given flow (normal or expedited) usually are processed sequentially within the path, but the expedited flow traffic can be moved ahead of the normal flow traffic within the path at queuing points in the half sessions and for half session support in boundary functions.

Notify
Used to synchronize awareness between SSCP and PU of status of a cross network session.

NS
See network services.

NS-IPL-ABORT
NS IPL abort.

NS-IPL-FINAL
NS IPL final.
NS-IPL-INIT
NS IPL initial.

NS-IPL-TEXT
NS IPL text.

NS-LSA
NS lost subarea.

NSPE
NS procedure error.

OAF
See origin address field.

OAF’
See origin address field prime.

OEF
See origin element field.

Origin Address Field (OAF)
A field in an FID0 or FID1 transmission header that contains the address of the originating network addressable unit. Contrast with destination address field. See also format identification (FID) field, local session identification (LSID), origin address field prime (OAF’), origin element field (OEF), origin subarea field (OSAF).

Origin Element Field (OEF)
A field in an FID4 transmission header that contains an element address, which combined with the subarea address in the origin subarea field (OSAF), gives the complete network address of the originating network addressable unit (NAU). Contrast with destination element field (DEF). See also format identification (FID) field.

Origin Subarea Field (OSAF)
A field in an FID4 transmission header that contains a subarea address, which combined with the element address in the origin element field (OEF), gives the complete network address for the originating network addressable unit (NAU). Contrast with destination subarea field (OSAF). See also format identification (FID) field.

OSAF
See origin subarea field.

Pacing
A technique by which a receiving component controls the rate of transmission of a sending component to prevent overrun or congestion.

Pacing Group
1. The path information units (PIUs) that can be transmitted on a virtual route before a virtual route pacing response is received, indicating that the virtual route receiver is ready to accept more PIUs on the route.
2. The requests that can be transmitted on the normal flow in one direction in a session before a session level pacing response is received, indicating that the receiver is ready to accept the next group of requests.
3. Synonymous with window.

Pacing Response
An indicator that signifies a receiving component's readiness to accept another pacing group; the indicator is carried in a response header (RH) for session level pacing, and in a

Parallel Sessions
Two or more concurrently active sessions between the same two logical units (LUs) using different pairs of network addresses. Each session can have independent session parameters.

Path
The series of path control network components (path control and data link control) that are traversed by the information exchanged between two network addressable units (NAUs). A path consists of a virtual route extension, if any. See also explicit route.

Path Control (PC) Layer
The layer that manages the sharing of link resources of the SNA network and routes basic information units (BIU) through it. Path control routes message units between network addressable units (NAUs) in the network and provides the paths between them. It converts the BIUs from transmission control (possibly segmenting them) into path information units (PIU) and exchanges basic transmission units (BTUs) – one or more PIUs – with data link control. See also BIU segment, data link control layer, transmission control (TC) layer.

NOTE
The unit of control information built by the sending path control component is the transmission header (TH), attached to the BTU; the TH is interpreted by the receiving path control component. The path control layer in subarea nodes consists of explicit route control, transmission group control, virtual route control, and boundary function path control.
**Path Information Unit (PIU)**
A message unit consisting of a transmission header (TH) alone, or of a TH followed by a basic information unit (BIU) or a BIU segment. See also transmission header.

**PC**
Path control.

**PDI**
Padded data indicator.

**P/F**
SDLC poll/final bit.

**Physical Unit (PU)**
The component that manages and monitors the resources (such as attached links and adjacent link stations) of a node, as requested by an SSCP via an SSCP–PU session. Each node of an SNA network contains a physical unit.

**NOTE**
An SSCP activates a session with the physical unit in order to indirectly manage, through the PU, resources of the node such as attached links and adjacent link stations.

**PI**
Pacing indicator. See also pacing response.

**PIU**
See path information unit.

**Point To Point Line**
A link that connects a single remote link station to a node; it can be switched or non-switched.

**Positive Response**
A response indicating that a request was successfully received and processed. Contrast with negative response.

**Presentation Services (PS)**
See session presentation services.

**PRI**
See primary link station.

**Primary Half Session**
The half session that sends the session activation request. Contrast with secondary half session.

**Primary Link Station (PRI)**
The link station on a link that is responsible for the control of that link. A link has only one primary link station. All traffic over the link is between the primary link station and a secondary link station. Contrast with secondary link station.

**PROCSTAT**
Procedure status.

**Protocol**
The meanings of, and the sequencing rules for, requests and responses used for managing the network, transferring data, and synchronizing the states of network components.

**PS**
See session presentation services.

**PU**
See physical unit.

**Public Network**
A network established and operated by communication common carriers or telecommunication administrations for the specific purpose or providing circuit switched, packet switched, and leased circuit services to the public.

**PU–PU Flow**
The exchange between physical units (PUs) of network control requests and responses.

**QC**
Quiesce complete.

**QEC**
Quiesce at end of chain.

**RD**
SDLC request disconnect frame.

**Receive Pacing**
The pacing of message units that a component is receiving. See also send pacing.

**RECFMS**
Record formatted maintenance statistics.

**RECMS**
Record maintenance statistics.

**RECSTOR**
Record storage.

**RECTD**
Record test data.
RECTR
Record test results.

RECTRD
Record trace data.

REJ
SDLC reject frame.

RELQ
Release queues.

Reply
A request unit sent only in reaction to a received request unit. For example, Quiesce Complete is the reply sent after receipt of Quiesce At End of Chain.

REQACTLU
Request activate logical unit.

REQCONT
Request contact.

REQDISCONT
Request discontact.

REQECHO
Request echo test.

REQFNA
Request free network address.

REQMS
Request maintenance statistics.

Request
A message unit that signals initiation of a particular action or protocol. For example, INITIATE SELF is a request for activation of an LU-LU session.

Request Header (RH)
A request unit (RU) header preceding a request unit.

Request Unit (RU)
A message unit that contains control information such as a request code, or function management (FM) headers, end user data, or both.

Request/Response Header (RH)
Control information, preceding a request/response unit (RU), that specifies the type of RU (request unit or response unit) and contains control information associated with that RU.

Request/Response Unit (RU)
A generic term for a request unit or a response unit.

Response
1. A message unit that acknowledges receipt of a request; a response consists of a response header (RH), a response unit (RU), or both.
2. In SDLC, the control information (in the C-Field of the link header) sent from the secondary station to the primary station.

Response Header (RH)
A header, optionally followed by a response unit (RU), that indicates whether the response is positive or negative and that it can contain a pacing response.

Response Unit (RU)
A message unit that acknowledges a request unit; it can contain prefix information received in a request unit. If positive, the response unit can contain additional information (such as session parameters in response to BIND SESSION), or if negative, sense data defining the exception condition.

RH
See request/response header, request header, response header.

RIM
SDLC request initialization mode frame.

Route
See explicit route, virtual route.

RJE
Remote job entry.

RNAA
Request network address assignment.

RNR
SDLC receive not ready frame.

ROUTE-TEST
Route test.

RPO
Remote power off.

RQR
Request recovery.

RR
SDLC receive ready frame.

RSHUTD
Request shutdown.
RTI
Response type indicator.

RTR
Ready to receive.

RU
See request/response unit, request unit, response unit.

RU Chain
A set of related request/response units (RUs) that are consecutively transmitted on a particular normal or expedited data flow. The request RU chain is the unit of recovery: if one of the RUs in the chain cannot be processed, the entire chain is discarded.

NOTE
Each RU belongs to only one chain, which has a beginning and an end indicated via control bits in request/response headers within the RU chain. Each RU can be designated as first-in-chain (FIC), last-in-chain (LIC), middle-in-chain (MIC), or only-in-chain (OIC). Response units and expedited flow request units are always sent as only-in-chain.

SBI
Stop bracket initiation.

SC
See session control.

SCS
See SNA character string.

SDI
Sense data included indicator.

SDLC
Synchronous data link control.

SDT
Start data traffic.

SEC
See secondary link station.

Secondary Half Session
The half session that receives the session activation request. Contrast with primary half session.

Secondary Link Station
Any link station on a link, using a primary–secondary protocol, that is not the primary link station. A secondary link station can exchange data only with the primary link station; no data traffic flows from one secondary link station to another. Contrast with primary link station.

Secondary Station
See secondary link station.

Segmenting of BIUs
An optional function of path control that divides a basic information unit (BIU) received from transmission control into two or more path information units (PIUs). The first PIU contains the request header (RH) of the BIU and usually part of the RU; the remaining PIUs contain the remaining parts of the RU.

NOTE
When segmenting is not done, a PIU contains a complete BIU.

Send Pacing
Pacing of message units that a component is sending. See also receive pacing.

SESEND
Session end.

Session
A logical connection between two network addressable units (NAUs) that can be activated, tailored to provide various protocols, and deactivated, as requested. The session activation request and response can determine options relating to such things as the rate and concurrency of data exchange, the control of contention and error recovery, and the characteristics of the data stream. Sessions compete for network resources such as the links within the path control network. See also half session, LU–LU session, SSCP–LU session, SSCP–PU session, SSCP–SSCP session.

NOTE
For routing purposes, each session is identified by the network (or local) address of the session partners.

Session Activation
The process of exchanging a session activation request and a positive response between network addressable units (NAUs). Contrast with session deactivation.
Session Activation Request
A request that activates a session between two network addressable units (NAUs) and specifies session parameters that control various protocols during session activity; for example, BIND and ACTPU. Contrast with session deactivation request.

Session Control (SC)
1. One of the components of transmission control. Session control is used to purge data flowing in a session after an unrecoverable error occurs, to resynchronize the data flow after such an error, and to perform cryptographic verification.
2. An RU category used for requests and responses exchanged between the session control components of a session and for session activation/deactivation requests and responses.

Session Count
1. The number of currently active LU-LU sessions for a particular logical unit.
2. The number of currently active sessions for a particular virtual route.

Session Deactivation
The process of exchanging a session deactivation request and response between network addressable units (NAUs). Contrast with session activation.

Session Deactivation Request
A request that deactivates a session between two network addressable units (NAUs); for example, UNBIND and DACTPU. Contrast with session activation request.

Session Initiation Request
An initiate or logon request from a logical unit (LU) to a system services control point (SSCP) that an LU-LU session be activated.

Session Limit
The maximum number of concurrently active LU-LU sessions a particular logical unit (LU) can support.

Session Level Pacing
A flow control technique that permits a receiving half session to control the data transfer rate (the rate at which it receives request units) on the normal flow. It is used to prevent overloading a receiver with unprocessed requests when the sender can generate requests faster than the receiver can process them. See also pacing, virtual route (VR) pacing.

Session Network Services
Network services that are performed on a half session by half session basis, rather than for the network addressable unit (NAU) as a whole.

Session Parameters
The parameters that specify or constrain the protocols (such as bracket protocol and pacing) for a session between two network addressable units (NAUs).

Session Partner
One of the two network addressable units (NAUs) having an active session.

Session Presentation Services (PS)
A component of the FMD services layer that provides, within LU-LU sessions, services for the application programmer or terminal operator such as formatting data to be displayed or printed.

Session Services
One of the types of network services in the system services control point (SSCP) and in a logical unit (LU). These services provide facilities for a logical unit (LU) or a network operator to request that the SSCP initiate or terminate sessions between logical units. See also configuration services, maintenance services, management services.

SESSST
Session started.

SETCV
Set control vector.

SHUTC
Shutdown complete.

SHUTD
Shutdown.

SIG
Signal.

SIM
SDLC set initialization mode frame.

SNA
Systems network architecture.

SNA Character String (SCS)
A data stream composed of EBCDIC controls, optionally intermixed with end user data, that is carried within a request/response unit.
SDLC/SNA TERMINOLOGY

SNA Network
The part of a user application network that conforms to the formats and protocols of Systems Network Architecture. It enables reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network addressable units (NAUs), boundary function components, and the path control network.

SNA Node
A node that supports SNA protocols.

SNA Station
A station that supports SNA protocols.

SNA Terminal
A terminal that supports SNA protocols.

SNF
Sequence number field.

SSCP
See system services control point.

SSCP ID
A number that uniquely identifies a system services control point (SSCP). The SSCP ID is used in session activation requests sent to physical units (PUs) and to other SSCPs.

SSCP-LU Session
A session between a system services control point (SSCP) and a logical unit (LU); the session enables the LU to request the SSCP to help initiate LU-LU sessions.

SSCP-PU Session
A session between a system services control point (SSCP) and a physical unit (PU); SSCP-PU sessions allow SSCPs to send requests to and receive status information from individual nodes in order to control the network configuration.

SSCP Services
The components within a system services control point (SSCP) that provide configuration, maintenance, management, network, and session services for SSCP-LU, SSCP-PU and SSCP-SSCP sessions.

SSCP Services Manager
An SNA component that provides network services for all the half sessions of the system services control point (SSCP).

SSCP-SSCP Session
A session between the system services control point (SSCP) in one domain and the SSCP in another domain. An SSCP-SSCP session is used to initiate and terminate cross domain LU-LU sessions.

Station
1. A link station.
2. One or more computers, terminals, devices, and associated programs at a particular location.

STSN
Set test and test sequence numbers.

Synchronous Data Link Control (SDLC)
A discipline for managing synchronous, code-transparent, serial by bit information transfer over a link connection. Transmission exchanges can be full duplex or half duplex over switched or nonswitched links. The configuration of the link connection can be point to point, multipoint, or loop. SDLC conforms to subsets of the Advanced Data Communication Control Procedures (ADCCP) of the American National Standards Institute and High-Level Data Link Control (HDLC) of the International Standards Organization.

System Services Control Point (SSCP)
A focal point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other session services for end users of the network. Multiple SSCPs, cooperating as peers with one another, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its own domain.

Systems Network Architecture (SNA)
The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

NOTE
The purpose of the layered structure of SNA is to allow the ultimate origins and destinations of information — that is, the end users — to be independent of, and unaffected by, the way in which the specific SNA network services and facilities used for information exchange are provided.
TC
See transmission control layer.

TERM-OTHER
Terminate other.

TERM-OTHER-CD
Terminate other cross-domain.

TERM-SELF
Terminate self.

Terminal Node
A peripheral node that is not user-programmable, having less processing capability than a cluster controller node. Examples are the IBM 3277, 3767, 3614, and 3624.

TESTMODE
Test mode.

TH
See transmission header.

Transmission Control (TC) Layer
The layer within a half session that synchronizes and paces session level data traffic, checks session sequence numbers of requests, and enciphers and deciphers end user data. Transmission control has two components: the connection point manager and session control.

Transmission Header (TH)
Control information, optionally followed by a basic information unit (BIU) or a BIU segment, that is created and used by path control to route message units and to control their flow within the network. See also path information units.

UA
SDLC unnumbered acknowledgment frame.

UI
SDLC unnumbered information frame.

UNBIND
Unbind session.

UNBINDF
Unbind failure.

UP
SDLC unnumbered poll.

User Application Network
A configuration of data processing products (such as processors, controllers, and terminals) established and operated by users for the purpose of data processing or information exchange, which can use services offered by common carriers or telecommunication administrations. Contrast with public network.

Virtual Route (VR)
A logical connection:
1. Between two subarea nodes that is physically realized as a particular explicit route.
2. That is contained wholly within a subarea node for intranode sessions. A virtual route between distinct subarea nodes imposes a transmission priority on the underlying explicit route, provides flow control through virtual route pacing, and provides data integrity through sequence numbering of path information units (PIUs). See also explicit route (ER), path.

Virtual Route (VR) Pacing
A flow control technique used by the virtual route control component of path control at each end of a virtual route to control the rate at which path information units (PIUs) flow over the virtual route. VR pacing can be adjusted according to traffic congestion in any of the nodes along the route. See also pacing, session level pacing.

VR
See virtual route.

VR-INOP
Virtual route inoperative.

Window
Synonym for pacing group.

Window Size
Synonym for pacing group size.

XID
SDLC exchange identification frame.
<table>
<thead>
<tr>
<th>Command</th>
<th>Stack Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC_ERROR?</td>
<td>( -- flag )</td>
<td>Detects a frame with a CRC error</td>
</tr>
<tr>
<td>ABORT?</td>
<td>( -- flag )</td>
<td>Detects an abort on the line</td>
</tr>
<tr>
<td>?RTS_ON, ?RTS_OFF</td>
<td>( -- flag )</td>
<td>Detects a transition on the request to send lead</td>
</tr>
<tr>
<td>?CTS_ON, ?CTS_OFF</td>
<td>( -- flag )</td>
<td>Detects a transition on the clear to send lead</td>
</tr>
<tr>
<td>?DSR_ON, ?DSR_ON</td>
<td>( -- flag )</td>
<td>Detects a transition on the data set ready lead</td>
</tr>
<tr>
<td>?CD_ON, ?CD_OFF</td>
<td>( -- flag )</td>
<td>Detects a transition on the carrier detect lead</td>
</tr>
<tr>
<td>?DTR_ON, ?DTR_OFF</td>
<td>( -- flag )</td>
<td>Detects a transition on the data terminal ready lead</td>
</tr>
<tr>
<td>?SQ_ON, ?SQ_OFF</td>
<td>( -- flag )</td>
<td>Detects a transition on the signal quality lead</td>
</tr>
<tr>
<td>?RI_ON, ?RI_OFF</td>
<td>( -- flag )</td>
<td>Detects a transition on the ring indication lead</td>
</tr>
</tbody>
</table>

Table D-1 Physical Events

<table>
<thead>
<tr>
<th>Command</th>
<th>Stack Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS_ON, RTS_OFF</td>
<td>( -- )</td>
<td>Sets the request to send lead (V.28, V.35, V.36)</td>
</tr>
<tr>
<td>CTS_ON, CTS_OFF</td>
<td>( -- )</td>
<td>Sets the clear to send lead (V.28, V.35, V.36)</td>
</tr>
<tr>
<td>DSR_ON, DSR_ON</td>
<td>( -- )</td>
<td>Sets the data set ready lead (V.28, V.35, V.36)</td>
</tr>
<tr>
<td>CD_ON, CD_OFF</td>
<td>( -- )</td>
<td>Sets the carrier detect lead (V.28, V.35) and sets the channel received line signal detector lead (V.36)</td>
</tr>
<tr>
<td>DTR_ON, DTR_OFF</td>
<td>( -- )</td>
<td>Sets the data terminal ready lead (V.28, V.35, V.36)</td>
</tr>
<tr>
<td>SQ_ON, SQ_OFF</td>
<td>( -- )</td>
<td>Sets the signal quality lead (V.28, V.35, V.36)</td>
</tr>
<tr>
<td>RI_ON, RI_OFF</td>
<td>( -- )</td>
<td>Sets the ring indication lead (V.28, V.35) calling indicator (V.36)</td>
</tr>
<tr>
<td>DRS_ON, DRS_OFF</td>
<td>( -- )</td>
<td>Sets the data signal rate select lead</td>
</tr>
</tbody>
</table>

Table D-2 Setting Leads
### Frame Identifiers

<table>
<thead>
<tr>
<th>Frame Type</th>
<th>R*RR</th>
<th>R*RNR</th>
<th>R*REJ</th>
<th>R*REJ</th>
<th>R*UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>R*SNRM</td>
<td>R*DISC</td>
<td>R*SIM</td>
<td>R*SIM</td>
<td>R*TEST</td>
<td></td>
</tr>
<tr>
<td>R*XID</td>
<td>R*UA</td>
<td>R*RD</td>
<td>R*RIM</td>
<td>R*DM</td>
<td></td>
</tr>
<tr>
<td>R*UP</td>
<td>R*FRMR</td>
<td>R*I</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table D-3 Frame Identifiers**

### Frame Events

<table>
<thead>
<tr>
<th>Command</th>
<th>Stack Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?RX_BLU</td>
<td>( frame identifier -- flag )</td>
<td>Detects a frame or BLU</td>
</tr>
<tr>
<td>?RX_FRAME</td>
<td>( id#1</td>
<td>--</td>
</tr>
<tr>
<td>?ADDRESS</td>
<td>( PU address -- flag )</td>
<td>Detects a frame with a specific address</td>
</tr>
<tr>
<td>FRAME?</td>
<td>( -- )</td>
<td>Detects any type of frame</td>
</tr>
<tr>
<td>DTE?</td>
<td>( -- )</td>
<td>Detects any type of frame from the DTE</td>
</tr>
<tr>
<td>DCE?</td>
<td>( -- )</td>
<td>Detects any type of frame from the DCE</td>
</tr>
<tr>
<td>?MATCH_I</td>
<td>( &quot; string&quot; -- flag )</td>
<td>Detects a specific string in an I frame</td>
</tr>
<tr>
<td>?SEARCH_I</td>
<td>( &quot; string&quot; -- flag )</td>
<td>Detects a specific string anywhere in an I frame</td>
</tr>
</tbody>
</table>

**Table D-4 Frame Events**
## Table D-5  SNA Header Identifiers

<table>
<thead>
<tr>
<th>TH constants</th>
<th>R*FID0</th>
<th>R*FID1</th>
<th>R*FID2</th>
<th>R*FID3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping field constants</td>
<td>R*MIDDLE-SEGMENT</td>
<td>R*FIRST-SEGMENT</td>
<td>R*LAST-SEGMENT</td>
<td></td>
</tr>
<tr>
<td>R*WHOLE-BIU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RH constants</td>
<td>R*FMD</td>
<td>R*DFC</td>
<td>R*NC</td>
<td>R*SC</td>
</tr>
<tr>
<td>RU constants (SC) category</td>
<td>R*ACTDRM</td>
<td>R*ACTLUC</td>
<td>R*ACTPU</td>
<td>R*BIND</td>
</tr>
<tr>
<td>R*CLEAR</td>
<td>R*CRV</td>
<td>R*DACTCDRM</td>
<td>R*DACTLU</td>
<td></td>
</tr>
<tr>
<td>R*DACTPU</td>
<td>R*RQR</td>
<td>R*SDT</td>
<td>R*STSN</td>
<td></td>
</tr>
<tr>
<td>R*UNBIND</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RU constants (DFC) category</td>
<td>R*BID</td>
<td>R*BIS</td>
<td>R*CANCEL</td>
<td>R*CHASE</td>
</tr>
<tr>
<td>R*LUSTAT</td>
<td>R*QC</td>
<td>R*QEC</td>
<td>R*RELO</td>
<td></td>
</tr>
<tr>
<td>R*RSHTUD</td>
<td>R*RTR</td>
<td>R*SBI</td>
<td>R*SHUTC</td>
<td></td>
</tr>
<tr>
<td>R*SHUTD</td>
<td>R*SIG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RU constants (NC) category</td>
<td>R*RAUSC</td>
<td>R*LSA</td>
<td>R*NC-ACTVR</td>
<td>R*NC-DACTVR</td>
</tr>
<tr>
<td>R*NC-ER-ACT</td>
<td>R*NC-ER-TEST-REPLY</td>
<td>R*NC-ER-INOP</td>
<td>R*NC-ER-OP</td>
<td></td>
</tr>
<tr>
<td>R*NC-ER-TEST</td>
<td>R*NC-IPL-FINAL</td>
<td>R*NC-ER-ACT-REPLY</td>
<td>R*NC-IPL-INIT</td>
<td></td>
</tr>
<tr>
<td>R*NC-IPL-TEXT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RU constants (FMD) category</td>
<td>R*ABCNOON</td>
<td>R*ABCNOTOUT</td>
<td>R*ACTCONN</td>
<td>R*ACTLINK</td>
</tr>
<tr>
<td>R*ACTTRACE</td>
<td>R*ADDLINK</td>
<td>R*ADDLINKSTA</td>
<td>R*ANA</td>
<td></td>
</tr>
<tr>
<td>R*BINDF</td>
<td>R*CDCINIT</td>
<td>R*CINIT</td>
<td>R*CDSESSEND</td>
<td></td>
</tr>
<tr>
<td>R*CDSESSSF</td>
<td>R*CDSESST</td>
<td>R*CDSESSTF</td>
<td>R*CDTAKED</td>
<td></td>
</tr>
<tr>
<td>R*CDTAKEDC</td>
<td>R*CDTERM</td>
<td>R*CESLOW</td>
<td>R*CEXIT</td>
<td></td>
</tr>
<tr>
<td>R*CLEANUP</td>
<td>R*CONNOUT</td>
<td>R*CONTACT</td>
<td>R*CONTACTED</td>
<td></td>
</tr>
<tr>
<td>R*CTERM</td>
<td>R*DACTCONN</td>
<td>R*DACTLINK</td>
<td>R*DACTRACE</td>
<td></td>
</tr>
<tr>
<td>R*DELETE</td>
<td>R*DELIVER</td>
<td>R*DISCONTACT</td>
<td>R*DISPSTOR</td>
<td></td>
</tr>
<tr>
<td>R*DSRSLT</td>
<td>R*DUMPFINAL</td>
<td>R*DUMPINIT</td>
<td>R*DUMPTEXT</td>
<td></td>
</tr>
<tr>
<td>R*ECHOTEST</td>
<td>R*ER-INOP</td>
<td>R*ER-TESTED</td>
<td>R*ESLOW</td>
<td></td>
</tr>
<tr>
<td>R*EXCETEST</td>
<td>R*EXSWLOW</td>
<td>R*FNA</td>
<td>R*FORWARD</td>
<td></td>
</tr>
<tr>
<td>R*INIT-OTHER</td>
<td>R*INIT-OTHER-CD</td>
<td>R*INIT-SELF_F0</td>
<td>R*INIT-SELF_F1</td>
<td></td>
</tr>
<tr>
<td>R*INITPROC</td>
<td>R*INOP</td>
<td>R*IPLFINAL</td>
<td>R*IPINIT</td>
<td></td>
</tr>
<tr>
<td>R*IPLTEXT</td>
<td>R*ISETCVR</td>
<td>R*LCP</td>
<td>R*LDEQD</td>
<td></td>
</tr>
<tr>
<td>R*NMTN</td>
<td>R*NOTIFY_SSCP-LU</td>
<td>R*NS-IPL-ABORT</td>
<td>R*NS-IPL-FINAL</td>
<td></td>
</tr>
<tr>
<td>R*NS-IPL-INIT</td>
<td>R*NS-IPL-TEXT</td>
<td>R*NS-LSA</td>
<td>R*NSL</td>
<td></td>
</tr>
<tr>
<td>R*NSIP</td>
<td>R*PROCSTAT</td>
<td>R*RECFSMS</td>
<td>R*RECM</td>
<td></td>
</tr>
<tr>
<td>R*RECMU</td>
<td>R*RECSR</td>
<td>R*RECTD</td>
<td>R*RECTR</td>
<td></td>
</tr>
<tr>
<td>R*RECTRD</td>
<td>R*REQACTLU</td>
<td>R*REQCONT</td>
<td>R*REODISCONT</td>
<td></td>
</tr>
<tr>
<td>R*REOECHO</td>
<td>R*REOFNA</td>
<td>R*REOMS</td>
<td>R*REOTEST</td>
<td></td>
</tr>
<tr>
<td>R*RNAA</td>
<td>R*ROUTE-TEST</td>
<td>R*RPO</td>
<td>R*SEESSEND</td>
<td></td>
</tr>
<tr>
<td>R*SESST</td>
<td>R*SETCT-V</td>
<td>R*SETCT-MAINT</td>
<td>R*STARTMEAS</td>
<td></td>
</tr>
<tr>
<td>R*TERM-OTHER</td>
<td>R*TERM-OTHER-CD</td>
<td>R*TERM-SELF</td>
<td>R*TERM-SELF_F1</td>
<td></td>
</tr>
<tr>
<td>R*TESTMODE</td>
<td>R*UNBINDF</td>
<td>R*VR-INOP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SNA Header Events

<table>
<thead>
<tr>
<th>Command</th>
<th>Stack Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?THAF</td>
<td>(address field -- flag)</td>
<td>Detects a specific origination or destination address field</td>
</tr>
<tr>
<td>?MAPPING_FIELD</td>
<td>(mapping field -- flag)</td>
<td>Detects a specific mapping field</td>
</tr>
<tr>
<td>?RX_REQ</td>
<td>(-- flag)</td>
<td>Detects a request type response/request header</td>
</tr>
<tr>
<td>?RX_RES</td>
<td>(-- flag)</td>
<td>Detects a response type response/request header</td>
</tr>
<tr>
<td>?RX_RH</td>
<td>(RU category -- flag)</td>
<td>Detects a specific request/response header</td>
</tr>
<tr>
<td>?RX_RU</td>
<td>(RU value -- flag)</td>
<td>Detects a specific response unit header</td>
</tr>
</tbody>
</table>

**Table D-6 SNA Header Events**

### Sending Frames and BTUs

<table>
<thead>
<tr>
<th>Command</th>
<th>Stack Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENDF</td>
<td>(&quot;string&quot; --)</td>
<td>Sends a string as a frame</td>
</tr>
<tr>
<td>SEND_BTU</td>
<td>(&quot;string&quot; --)</td>
<td>Sends a string as a BTU</td>
</tr>
<tr>
<td>SEND_BUFFER</td>
<td>(buffer number --)</td>
<td>Sends a buffer as an I frame</td>
</tr>
<tr>
<td>SNRM , S:SNRM</td>
<td>(-- )</td>
<td>Sends a SNRM frame</td>
</tr>
<tr>
<td>DISC, S:DISC</td>
<td>(-- )</td>
<td>Sends a DISC frame</td>
</tr>
<tr>
<td>SIM, S:SIM</td>
<td>(-- )</td>
<td>Sends a SIM frame</td>
</tr>
<tr>
<td>XID, S:XID</td>
<td>(-- )</td>
<td>Sends a XID frame</td>
</tr>
<tr>
<td>RIM, S:RIM</td>
<td>(-- )</td>
<td>Sends a RIM frame</td>
</tr>
<tr>
<td>RM, S:RM</td>
<td>(-- )</td>
<td>Sends a RM frame</td>
</tr>
<tr>
<td>S:UA</td>
<td>(-- )</td>
<td>Sends a UA frame</td>
</tr>
<tr>
<td>S:DM</td>
<td>(-- )</td>
<td>Sends a DM frame</td>
</tr>
<tr>
<td>S:UP</td>
<td>(-- )</td>
<td>Sends a UP frame</td>
</tr>
<tr>
<td>S:RR</td>
<td>(-- )</td>
<td>Sends a RR frame</td>
</tr>
<tr>
<td>S:RNR</td>
<td>(-- )</td>
<td>Sends a RNR frame</td>
</tr>
<tr>
<td>S:REJ</td>
<td>(-- )</td>
<td>Sends a reject frame</td>
</tr>
<tr>
<td>BUILD_TH_BYTE0</td>
<td></td>
<td>Builds byte 0 of the TH to send using SEND_BUFFER</td>
</tr>
<tr>
<td>BUILD_TH</td>
<td>(string -- string)</td>
<td>Builds the TH to send using SEND_BTU</td>
</tr>
<tr>
<td>BUILD_RH</td>
<td>(string -- string)</td>
<td>Builds RH to send using SEND_BTU</td>
</tr>
<tr>
<td>BUILD_RU</td>
<td>(string -- string)</td>
<td>Builds RU to send using SEND_BTU</td>
</tr>
</tbody>
</table>

**Table D-7 Sending Frames and BTU's**
### Creating Buffers

<table>
<thead>
<tr>
<th>Command</th>
<th>Stack Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE-&gt;BUFFER</td>
<td>( filename\buffer number -- )</td>
<td>Loads a buffer from a file</td>
</tr>
<tr>
<td>STRING-&gt;BUFFER</td>
<td>( string\buffer number -- )</td>
<td>Loads a buffer from a string (maximum 255 bytes)</td>
</tr>
<tr>
<td>ALLOT_BUFFER</td>
<td>( size\buffer number -- flag )</td>
<td>Allocates memory for a buffer</td>
</tr>
<tr>
<td>FILL_BUFFER</td>
<td>( data address\size\buffer number -- )</td>
<td>Moves data into a buffer and overwrites the previous contents</td>
</tr>
<tr>
<td>APPEND_TO_BUFFER</td>
<td>( data address\size\buffer number -- )</td>
<td>Appends data into a buffer</td>
</tr>
<tr>
<td>CLEAR_BUFFER</td>
<td>( buffer number -- )</td>
<td>Stores a size of 0 in the buffer</td>
</tr>
<tr>
<td>BUFFER</td>
<td>( buffer number -- address )</td>
<td>Returns the address of the first byte of the specified buffer</td>
</tr>
</tbody>
</table>

**Table D–8 Creating Buffers**

### Starting and Examining Timers

<table>
<thead>
<tr>
<th>Command</th>
<th>Stack Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>START_TIMER</td>
<td>( timer#\time-- )</td>
<td>Starts an alarm (countdown) timer</td>
</tr>
<tr>
<td>STOP_TIMER</td>
<td>( timer# -- )</td>
<td>Stops (resets) an alarm timer</td>
</tr>
<tr>
<td>START_LAPSE_TIMER</td>
<td>( timer# -- )</td>
<td>Starts an elapsed time timer</td>
</tr>
<tr>
<td>MINUTES_ELAPSED</td>
<td>( timer# -- minutes )</td>
<td>Examines the minutes elapsed for elapsed time timer</td>
</tr>
<tr>
<td>SECONDS_ELAPSED</td>
<td>( timer# -- seconds )</td>
<td>Examines the seconds elapsed for elapsed time timer</td>
</tr>
<tr>
<td>MILLISECONDS_ELAPSED</td>
<td>( timer# -- milliseconds )</td>
<td>Examines the milliseconds elapsed for elapsed time timer</td>
</tr>
</tbody>
</table>

**Table D–9 Starting and Examining Timers**

### Timer Events

<table>
<thead>
<tr>
<th>Command</th>
<th>Stack Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMEOUT</td>
<td>( -- flag )</td>
<td>Detects a timeout of any user timer</td>
</tr>
<tr>
<td>?TIMER</td>
<td>( n -- flag )</td>
<td>Detects a timeout of a specific user timer</td>
</tr>
<tr>
<td>?WAKEUP</td>
<td>( -- flag )</td>
<td>Detects wakeup timer</td>
</tr>
</tbody>
</table>

**Table D–10 Timer Events**
<table>
<thead>
<tr>
<th>Command</th>
<th>Stack Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T.&quot; goes to RAM and Disk too!&quot;</td>
<td>( -- ) 1 space required after T.&quot;</td>
<td>Displays a timestamped comment (trace statement) in the Data Window</td>
</tr>
<tr>
<td>TCR</td>
<td>( -- )</td>
<td>Inserts a carriage return with the trace statement</td>
</tr>
<tr>
<td>T.</td>
<td>( value -- )</td>
<td>Displays a decimal value in the Data Window</td>
</tr>
<tr>
<td>T.H</td>
<td>( value -- )</td>
<td>Displays a hexadecimal value in the Data Window</td>
</tr>
<tr>
<td>P.&quot; goes to the printer&quot;</td>
<td>( -- ) 1 space required after the P.&quot;</td>
<td>Prints a comment</td>
</tr>
<tr>
<td>PCR</td>
<td>( -- )</td>
<td>Sends a carriage return to the printer</td>
</tr>
<tr>
<td>P.</td>
<td>( value -- )</td>
<td>Prints a decimal value</td>
</tr>
<tr>
<td>P.H</td>
<td>( value -- )</td>
<td>Prints a hexadecimal value</td>
</tr>
</tbody>
</table>

Table D–11 Creating User Output
### Program Control Events

<table>
<thead>
<tr>
<th>Command</th>
<th>Stack Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?KEY</td>
<td>( user function key # -- )</td>
<td>Detects a function key</td>
</tr>
<tr>
<td>PROMPT&quot; text&quot; actions to be taken using string at address= prompt END_PROMPT</td>
<td>( -- )</td>
<td>Prompts the user for keyboard input</td>
</tr>
<tr>
<td>?MAIL</td>
<td>( -- flag )</td>
<td>Detects a signal from another ITL program</td>
</tr>
</tbody>
</table>

**Table D–12 Program Control Events**
The following 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.

**E.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 left-most item representing the bottom of the stack and the right-most 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 or a 1 is returned on the stack after the '=' operation, with 0 being a false result, and 1 a true result.
E.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 E-1 ITL Symbols

E.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"
   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.
E.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. Furthermore, the use of many meaningful comments is highly recommended and will 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 )
        ENDIF
    }ACTION
}STATE
```

E.5 Spacing and Indentation Guidelines

The following 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.);
- tab for nested constructs;
- carriage return after colon definition and stack comment; and
- carriage return after last line of code in colon definition and semi-colon;

See the examples in Appendices E.4 and E.6.
E.6 Colon Definitions

Colon definition should be preceded by a short comment. The colon definition should start at the first column of a line. All code 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
     ENDF
<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>68</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>69</td>
<td>9</td>
<td>IT</td>
</tr>
<tr>
<td>0A</td>
<td>10</td>
<td>10</td>
<td>LF</td>
<td>RPT</td>
<td>3A</td>
<td>58</td>
<td>70</td>
<td>:</td>
<td>RFF</td>
</tr>
<tr>
<td>0B</td>
<td>11</td>
<td>11</td>
<td>VT</td>
<td>VT</td>
<td>3B</td>
<td>59</td>
<td>71</td>
<td>;</td>
<td>CU3</td>
</tr>
<tr>
<td>0C</td>
<td>12</td>
<td>12</td>
<td>FF</td>
<td>FF</td>
<td>3C</td>
<td>60</td>
<td>72</td>
<td>&lt;</td>
<td>DC4</td>
</tr>
<tr>
<td>0D</td>
<td>13</td>
<td>13</td>
<td>CR</td>
<td>CR</td>
<td>3D</td>
<td>61</td>
<td>73</td>
<td>=</td>
<td>NAK</td>
</tr>
<tr>
<td>0E</td>
<td>14</td>
<td>14</td>
<td>SO</td>
<td>SO</td>
<td>3E</td>
<td>62</td>
<td>74</td>
<td>&gt;</td>
<td></td>
</tr>
<tr>
<td>0F</td>
<td>15</td>
<td>15</td>
<td>SI</td>
<td>SI</td>
<td>3F</td>
<td>63</td>
<td>75</td>
<td>?</td>
<td>SUB</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>20</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>21</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>22</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>23</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>24</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>25</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>26</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>27</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>30</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>31</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>32</td>
<td>SUB</td>
<td>UBS</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>33</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>34</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>35</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>36</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>37</td>
<td>US</td>
<td>IUS</td>
<td>4F</td>
<td>79</td>
<td>117</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td>40</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>41</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>42</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>43</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>44</td>
<td>$</td>
<td>BYP</td>
<td>54</td>
<td>84</td>
<td>124</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>37</td>
<td>45</td>
<td>%</td>
<td>LF</td>
<td>55</td>
<td>85</td>
<td>125</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>38</td>
<td>46</td>
<td>&amp;</td>
<td>ETB</td>
<td>56</td>
<td>86</td>
<td>126</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>39</td>
<td>47</td>
<td>'</td>
<td>ESC</td>
<td>57</td>
<td>87</td>
<td>127</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>40</td>
<td>50</td>
<td>(</td>
<td>SA</td>
<td>58</td>
<td>88</td>
<td>130</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>41</td>
<td>51</td>
<td>)</td>
<td>SFE</td>
<td>59</td>
<td>89</td>
<td>131</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>2A</td>
<td>42</td>
<td>52</td>
<td>*</td>
<td>SM/SW</td>
<td>5A</td>
<td>90</td>
<td>132</td>
<td>Z</td>
<td>!</td>
</tr>
<tr>
<td>2B</td>
<td>43</td>
<td>53</td>
<td>+</td>
<td>CSP</td>
<td>5B</td>
<td>91</td>
<td>133</td>
<td>[</td>
<td>$</td>
</tr>
<tr>
<td>2C</td>
<td>44</td>
<td>54</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>55</td>
<td>ENQ</td>
<td>5D</td>
<td>93</td>
<td>135</td>
<td>)</td>
<td>)</td>
<td></td>
</tr>
<tr>
<td>2E</td>
<td>46</td>
<td>56</td>
<td>ACK</td>
<td>5E</td>
<td>94</td>
<td>136</td>
<td>'</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>47</td>
<td>57</td>
<td>BEL</td>
<td>5F</td>
<td>95</td>
<td>137</td>
<td>-</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>l</td>
<td>93</td>
<td>147</td>
<td>223</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>100</td>
<td>144</td>
<td>d</td>
<td>m</td>
<td>94</td>
<td>148</td>
<td>224</td>
<td>n</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>o</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>p</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>q</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>r</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>s</td>
<td></td>
</tr>
<tr>
<td>6A</td>
<td>106</td>
<td>152</td>
<td>j</td>
<td></td>
<td>9A</td>
<td>154</td>
<td>232</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>6B</td>
<td>107</td>
<td>153</td>
<td>k</td>
<td></td>
<td>9B</td>
<td>155</td>
<td>233</td>
<td>u</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>v</td>
<td></td>
</tr>
<tr>
<td>6D</td>
<td>109</td>
<td>155</td>
<td>m</td>
<td></td>
<td>9D</td>
<td>157</td>
<td>235</td>
<td>w</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>x</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>y</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>z</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>.</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>]</td>
<td></td>
</tr>
<tr>
<td>7F</td>
<td>127</td>
<td>177</td>
<td></td>
<td>DEL</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></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></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></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></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></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></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></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></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></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></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></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></td>
<td></td>
<td>BB</td>
<td>187</td>
<td>273</td>
<td>J</td>
<td></td>
</tr>
<tr>
<td>8C</td>
<td>140</td>
<td>214</td>
<td></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></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></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></td>
<td></td>
<td>BF</td>
<td>191</td>
<td>277</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>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>F3</td>
<td>243</td>
<td>363</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>196</td>
<td>304</td>
<td>D</td>
<td></td>
<td>F4</td>
<td>244</td>
<td>364</td>
<td>4</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>J</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>K</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>L</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>M</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>N</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>O</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></td>
<td>J</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>210</td>
<td>322</td>
<td></td>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>211</td>
<td>323</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>212</td>
<td>324</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>213</td>
<td>325</td>
<td></td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td>214</td>
<td>326</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>215</td>
<td>327</td>
<td></td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D8</td>
<td>216</td>
<td>328</td>
<td></td>
<td>Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D9</td>
<td>217</td>
<td>330</td>
<td></td>
<td>R</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></td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td>227</td>
<td>343</td>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td>228</td>
<td>344</td>
<td></td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E5</td>
<td>229</td>
<td>345</td>
<td></td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E6</td>
<td>230</td>
<td>346</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E7</td>
<td>231</td>
<td>347</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E8</td>
<td>232</td>
<td>350</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E9</td>
<td>233</td>
<td>351</td>
<td></td>
<td>Z</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 and variables, not appearing in this manual, with new replacement commands. Reference should be made to the previous versions of this manual for description of the old commands. The new commands achieve the same function; however, the input/output parameters may have changed.

<table>
<thead>
<tr>
<th>Old Command</th>
<th>New Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFRMA=</td>
<td>DO_CFRMA</td>
</tr>
<tr>
<td>CTHAF=</td>
<td>DO_CTHAF</td>
</tr>
<tr>
<td>DATA-STATUS</td>
<td>STATUS_ERR?</td>
</tr>
<tr>
<td>DO_SETID</td>
<td>SET_LINK_ID</td>
</tr>
<tr>
<td>DFRMA=</td>
<td>DO_DFRMA</td>
</tr>
<tr>
<td>DTHAF=</td>
<td>DO_DTHAF</td>
</tr>
<tr>
<td>PLAY-ID</td>
<td>PORT-Id</td>
</tr>
<tr>
<td>PLAY-STIME</td>
<td>START-TIME</td>
</tr>
<tr>
<td>PLAY-ETIME</td>
<td>END-TIME</td>
</tr>
<tr>
<td>PLAY-COUNT</td>
<td>BLOCK-COUNT</td>
</tr>
<tr>
<td>POLL_TIME=</td>
<td>DO_PTIME</td>
</tr>
<tr>
<td>REC-STATUS/DATA_STATUS</td>
<td>STATUS_ERR?</td>
</tr>
<tr>
<td>RFRMA=</td>
<td>DO_RFRMA</td>
</tr>
<tr>
<td>RTHAF=</td>
<td>DO_RTHAF</td>
</tr>
<tr>
<td>SET_#LINKS=</td>
<td>DO_#LINKS</td>
</tr>
<tr>
<td>SET_ID=</td>
<td>SET_LINK_ID</td>
</tr>
<tr>
<td>SET_LINK#</td>
<td>DO_LINK#</td>
</tr>
<tr>
<td>SET_LONG</td>
<td>LONG-INTERVAL</td>
</tr>
<tr>
<td>SET_RETRY#</td>
<td>DO_SETRETRY</td>
</tr>
<tr>
<td>SET_SPEED</td>
<td>INTERFACE-SPEED</td>
</tr>
<tr>
<td>SET_SHORT</td>
<td>SHORT-INTERVAL</td>
</tr>
<tr>
<td>SET_WINDOW=</td>
<td>DO_SETW</td>
</tr>
</tbody>
</table>
INDEX

3274 Terminal
simulation, 14-18 to 14-20
testing, 14-10 to 14-20

Abort

test manager event, 13-6
transmitting, 12-17

ABORT?, 13-6

ACTION( )ACTION, 13-1

?ADDRESS, 13-5

ALLOT_BUFFER, 13-19

ALL_LEADS, 2-2, 9-3

APPEND_TO_BUFFER, 13-19

Application Layer, A-21

Architecture

eulation, 10-1 to 10-4
monitor, 3-1 to 3-4

ASCII, 7-4

Automatic Response, 12-3

B, see BACKWARD

BACKWARD, 3-3

BAD-CONTROL, 12-15

BLU, see Frame(s)

Begin Bracket Indicator, 4-12

Begin Chain Indicator, 4-11, 11-5

Bit Rate

emulation, 9-4
monitor, 2-3

Block Number

decode, 4-2
display format, 7-3

BLOCK-COUNT, 4-2

BUFFER, 13-20

Buffer(s), 13-18 to 13-21

allocating memory, 13-19
appending text, 13-19
clearing, 13-20
frame information field, 12-16
moving text, 13-19
number, 13-18
sending, 13-20, 13-21
size, 13-18
structure, 13-18

TEST frame, 12-15

UI frame information field, 12-16

BUFFER_SEND, 13-20

BUILD_RH, 11-7, 12-11, 12-13

BUILD_RU, 12-12

BUILD_TH, 12-11, 12-13

BUILD_TH–BYTE0, 12-10

BUSY, 12-7

BYTE-COUNT, 11-2

C1•ALL, 8-4

C1•NONE, 8-5

C2+, see Filters, frame layer

C2–, see Filters, frame layer

Capture RAM

capturing to RAM, 5-1, 5-2
clearing, 5-2
configuring, 5-2
playback, 3-2
printing, 5-4

saving to disk, 5-4

CAPT_FULL, 5-2

CAPT_OFF, 5-1

CAPT_ON, 5-1

CAPT_WRAP, 5-2

CFID+, see Filters, transmission header

CFID–, see Filters, transmission header

CFRM+, see Filters, frame layer

CFRM–, see Filters, frame layer

CFRMXING, 8-5

CHAIN-IND, 4-11

Change Direction Indicator, 4-12, 11-6

CHANGE-DIR-IND, 4-12

Character Set

ASCII, 7-4

EBCDIC, 7-4

hex, 7-4

JIS8, 7-4

CK_LEAD_ENABLE, 2-3

CLEAR_BUFFER, 13-20

CLEAR_CRT, 7-7

Clocking

external, 2-3, 9-4

NRZI, 2-3, 9-5

NRZI with clock, 2-3, 9-5

standard, 2-3, 9-4

Code Selection Indicator, 4-12, 11-6

CODE-SEL-IND, 4-12

COMMAND-FORMAT, 4-9

COMMAND-SEQUENCE-NUMBER, 4-10

COMMAND-TYPE, 4-9

COMMAND_IND, 13-16

Comparison

anchored, 13-13

unanchored, 13-14

wildcard, 13-13

Configuration

capture RAM, 5-2

eulation, 9-1 to 9-9

monitor, 2-1 to 2-4

point-to-point, A-2

printer, 5-4

Connectors

V.11, 2-1, 9-2

V.28, 2-1, 9-2

V.35, 2-1, 9-2

V.36, 2-2, 9-2

CONTROL, 11-2

Control Lead
decode, 4-3

enabling, 2-3

filters, 8-4, 8-5

turning on/off, 13-17

CRC Error(s)
test manager event, 13-6

transmitting, 12-17

CRC–TYPE, 12-17

CRC_ERROR?, 13-6

CRH+, see Filters, request/response header

CRH–, see Filters, request/response header

CSD+, see Filters, sense data

CSD–, see Filters, sense data

CTHAFXING, 8-7

CTOD_OFF, 5-4

CTOD_ON, 5-4

CTRACE, 8-3, 8-4

D1•ALL, 8-5

D1•NONE, 8-5

D2+, see Filters, frame layer

D2–, see Filters, frame layer

Data Count Field, 4-6

Data Field Display Format, 7-7

Data Flow Control, A-20

Data Link Control Layer, see Frame Layer

DATA–COUNT–FIELD, 4-6

DATA_CHAR, 7-7

DATA_HEX, 7-7

DATA_OFF, 7-7

DATA_ON, 7-7

SDLC/SNA Programmer's Manual
<table>
<thead>
<tr>
<th>Index Entry</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCE?, 13-6</td>
<td></td>
</tr>
<tr>
<td>Decode</td>
<td></td>
</tr>
<tr>
<td>block number, 4-2</td>
<td></td>
</tr>
<tr>
<td>emulation, 11-1 to 11-8</td>
<td></td>
</tr>
<tr>
<td>frame layer, 4-3, 4-4, 11-2</td>
<td></td>
</tr>
<tr>
<td>monitor, 4-1 to 4-15</td>
<td></td>
</tr>
<tr>
<td>physical layer, 4-2, 4-3</td>
<td></td>
</tr>
<tr>
<td>request/response header, 4-10 to 4-13</td>
<td></td>
</tr>
<tr>
<td>request/response unit, 4-14, 4-15</td>
<td></td>
</tr>
<tr>
<td>sense data, 4-13, 4-14</td>
<td></td>
</tr>
<tr>
<td>timestamp, 4-2</td>
<td></td>
</tr>
<tr>
<td>transmission header, 4-4 to 4-10, 11-3</td>
<td></td>
</tr>
<tr>
<td>DEF, 4-9</td>
<td></td>
</tr>
<tr>
<td>Default Response Indicator, 4-11, 11-5</td>
<td></td>
</tr>
<tr>
<td>DEST-ADD=FIELD, 4-6</td>
<td></td>
</tr>
<tr>
<td>Destination Address Field</td>
<td></td>
</tr>
<tr>
<td>decode, 4-5, 11-3</td>
<td></td>
</tr>
<tr>
<td>encode, 11-4</td>
<td></td>
</tr>
<tr>
<td>filters, 8-7, 8-8</td>
<td></td>
</tr>
<tr>
<td>test manager event, 13-7</td>
<td></td>
</tr>
<tr>
<td>user-defined, 12-8</td>
<td></td>
</tr>
<tr>
<td>Destination Element Field, 4-9</td>
<td></td>
</tr>
<tr>
<td>Destination Sub-area Address, 4-9</td>
<td></td>
</tr>
<tr>
<td>DFID+, see Filters, transmission header</td>
<td></td>
</tr>
<tr>
<td>DFID-, see Filters, transmission header</td>
<td></td>
</tr>
<tr>
<td>DFRM+, see Filters, frame layer</td>
<td></td>
</tr>
<tr>
<td>DFRM-, see Filters, frame layer</td>
<td></td>
</tr>
<tr>
<td>DFRMA=ALL, 8-6</td>
<td></td>
</tr>
<tr>
<td>DISABLE_LEAD, 2-2, 9-3</td>
<td></td>
</tr>
<tr>
<td>DISC, 12-6</td>
<td></td>
</tr>
<tr>
<td>DISC/UA_WAIT, 12-2</td>
<td></td>
</tr>
<tr>
<td>DISK_FILTER_OFF, 8-3</td>
<td></td>
</tr>
<tr>
<td>DISK_FILTER_ON, 8-3</td>
<td></td>
</tr>
<tr>
<td>DISK_FULL, 6-1</td>
<td></td>
</tr>
<tr>
<td>DISK_OFF, 6-2</td>
<td></td>
</tr>
<tr>
<td>DISK_WRAP, 6-1</td>
<td></td>
</tr>
<tr>
<td>Display Format, 7-1 to 7-10</td>
<td></td>
</tr>
<tr>
<td>character, 7-3</td>
<td></td>
</tr>
<tr>
<td>character set, 7-4</td>
<td></td>
</tr>
<tr>
<td>complete, 7-2</td>
<td></td>
</tr>
<tr>
<td>data field, 7-7</td>
<td></td>
</tr>
<tr>
<td>dual, 7-7</td>
<td></td>
</tr>
<tr>
<td>frame layer, 7-4</td>
<td></td>
</tr>
<tr>
<td>full, 7-7</td>
<td></td>
</tr>
<tr>
<td>hex, 7-3</td>
<td></td>
</tr>
<tr>
<td>request/response header, 7-5</td>
<td></td>
</tr>
<tr>
<td>request/response unit, 7-6</td>
<td></td>
</tr>
<tr>
<td>sense data, 7-6</td>
<td></td>
</tr>
<tr>
<td>short, 7-2</td>
<td></td>
</tr>
<tr>
<td>timestamp, 7-3</td>
<td></td>
</tr>
<tr>
<td>trace statements, 7-3, 7-9</td>
<td></td>
</tr>
<tr>
<td>transmission header, 7-5</td>
<td></td>
</tr>
<tr>
<td>DIS_REC, 6-2</td>
<td></td>
</tr>
<tr>
<td>DO#LINKS, 9-6</td>
<td></td>
</tr>
<tr>
<td>DO_CFMA, 8-6</td>
<td></td>
</tr>
<tr>
<td>DO_CHAF, 8-8</td>
<td></td>
</tr>
<tr>
<td>DO_DFRM, 8-7</td>
<td></td>
</tr>
<tr>
<td>DO_DTHAF, 8-8</td>
<td></td>
</tr>
<tr>
<td>DO_LINK#, 9-8</td>
<td></td>
</tr>
<tr>
<td>DO_PTME, 9-7</td>
<td></td>
</tr>
<tr>
<td>DO_RFMA, 8-6</td>
<td></td>
</tr>
<tr>
<td>DO_RTHAF, 8-8</td>
<td></td>
</tr>
<tr>
<td>DO_SETRETRY, 9-9</td>
<td></td>
</tr>
<tr>
<td>DO_SETTO, 9-9</td>
<td></td>
</tr>
<tr>
<td>DO_SETW, 9-9</td>
<td></td>
</tr>
<tr>
<td>DRH+, see Filters, request/response header</td>
<td></td>
</tr>
<tr>
<td>DRH-, see Filters, request/response header</td>
<td></td>
</tr>
<tr>
<td>DSAF, 4-9</td>
<td></td>
</tr>
<tr>
<td>DSD, see Filters, sense data</td>
<td></td>
</tr>
<tr>
<td>DSD-, see Filters, sense data</td>
<td></td>
</tr>
<tr>
<td>DTE#, 13-5</td>
<td></td>
</tr>
<tr>
<td>DTHAF=ALL, 8-7</td>
<td></td>
</tr>
<tr>
<td>DTRACE, 8-4</td>
<td></td>
</tr>
<tr>
<td>EBCDIC, 7-4</td>
<td></td>
</tr>
<tr>
<td>Emulation architecture, 10-1 to 10-4</td>
<td></td>
</tr>
<tr>
<td>automatic, 9-6, 12-3</td>
<td></td>
</tr>
<tr>
<td>bit rate, 9-4</td>
<td></td>
</tr>
<tr>
<td>clocking, 9-4</td>
<td></td>
</tr>
<tr>
<td>configuration, 9-1 to 9-9</td>
<td></td>
</tr>
<tr>
<td>decode, 11-1 to 11-8</td>
<td></td>
</tr>
<tr>
<td>disconnecting a link, 12-4, 12-5, 12-7</td>
<td></td>
</tr>
<tr>
<td>establishing a link, 12-3</td>
<td></td>
</tr>
<tr>
<td>initializing a link, 12-4</td>
<td></td>
</tr>
<tr>
<td>live data, 10-1, 10-2</td>
<td></td>
</tr>
<tr>
<td>manual, 9-6, 12-3</td>
<td></td>
</tr>
<tr>
<td>multipoint, 9-7</td>
<td></td>
</tr>
<tr>
<td>path, 10-1 to 10-4</td>
<td></td>
</tr>
<tr>
<td>playback, 10-3</td>
<td></td>
</tr>
<tr>
<td>point-to-point, 9-7</td>
<td></td>
</tr>
<tr>
<td>primary, 9-1</td>
<td></td>
</tr>
<tr>
<td>primary state machine, 12-1 to 12-3</td>
<td></td>
</tr>
<tr>
<td>response, 12-1 to 12-17</td>
<td></td>
</tr>
<tr>
<td>secondary, 9-1</td>
<td></td>
</tr>
<tr>
<td>secondary state machine, 12-2, 12-3</td>
<td></td>
</tr>
<tr>
<td>test script[s], 14-10 to 14-27</td>
<td></td>
</tr>
<tr>
<td>to DCE/DTE, 9-2</td>
<td></td>
</tr>
<tr>
<td>two-way alternate, 9-7</td>
<td></td>
</tr>
<tr>
<td>two-way simultaneous, 9-7</td>
<td></td>
</tr>
<tr>
<td>EMUL_OFF, 9-6, 12-3</td>
<td></td>
</tr>
<tr>
<td>EMUL_ON, 9-6, 12-3</td>
<td></td>
</tr>
<tr>
<td>ENABLE_LEAD, 2-2, 9-3</td>
<td></td>
</tr>
<tr>
<td>ENB_REC, 6-2</td>
<td></td>
</tr>
<tr>
<td>ENCRYPT=DATA-IND, 4-13</td>
<td></td>
</tr>
<tr>
<td>Enciphered Data Indicator, 4-13, 11-7</td>
<td></td>
</tr>
<tr>
<td>Encode</td>
<td></td>
</tr>
<tr>
<td>destination address field, 11-4</td>
<td></td>
</tr>
<tr>
<td>expedited flow indicator, 11-4</td>
<td></td>
</tr>
<tr>
<td>format identifier, 11-3</td>
<td></td>
</tr>
<tr>
<td>frame layer, 11-2, 11-3</td>
<td></td>
</tr>
<tr>
<td>mapping field, 11-3</td>
<td></td>
</tr>
<tr>
<td>origin address field, 11-4</td>
<td></td>
</tr>
<tr>
<td>request/response header, 11-4 to 11-7</td>
<td></td>
</tr>
<tr>
<td>request/response unit, 11-8</td>
<td></td>
</tr>
<tr>
<td>sense data, 11-7, 11-8</td>
<td></td>
</tr>
<tr>
<td>sequence number field, 11-4</td>
<td></td>
</tr>
<tr>
<td>transmission header, 11-3, 11-4</td>
<td></td>
</tr>
<tr>
<td>End Bracket Indicator, 4-12</td>
<td></td>
</tr>
<tr>
<td>End Chain Indicator, 4-11, 11-5</td>
<td></td>
</tr>
<tr>
<td>END-TIME, 4-2</td>
<td></td>
</tr>
<tr>
<td>ENTER, 12-2</td>
<td></td>
</tr>
<tr>
<td>EOF_IND, 13-16</td>
<td></td>
</tr>
<tr>
<td>ER-VR-SUPP-IND, 4-7</td>
<td></td>
</tr>
<tr>
<td>ERN, 4-7</td>
<td></td>
</tr>
<tr>
<td>Event Recognition, 13-2 to 13-16</td>
<td></td>
</tr>
<tr>
<td>abort, 13-6</td>
<td></td>
</tr>
<tr>
<td>anchored comparison, 13-13</td>
<td></td>
</tr>
<tr>
<td>CRC error, 13-6</td>
<td></td>
</tr>
<tr>
<td>destination address field, 13-7</td>
<td></td>
</tr>
<tr>
<td>frames, 13-3 to 13-14</td>
<td></td>
</tr>
<tr>
<td>from DCE/DTE, 13-6</td>
<td></td>
</tr>
<tr>
<td>mapping field, 13-8</td>
<td></td>
</tr>
<tr>
<td>origin address field, 13-7</td>
<td></td>
</tr>
<tr>
<td>physical layer, 13-3</td>
<td></td>
</tr>
<tr>
<td>PU, 13-5</td>
<td></td>
</tr>
<tr>
<td>request/response header, 13-8</td>
<td></td>
</tr>
<tr>
<td>request/response unit, 13-9</td>
<td></td>
</tr>
<tr>
<td>timers, 13-14, 13-15</td>
<td></td>
</tr>
<tr>
<td>transmission header, 13-7</td>
<td></td>
</tr>
<tr>
<td>unanchored comparison, 13-14</td>
<td></td>
</tr>
<tr>
<td>wildcard, 13-16</td>
<td></td>
</tr>
<tr>
<td>EVENT-TYPE, 13-16</td>
<td></td>
</tr>
<tr>
<td>Exception Response Indicator, 4-11, 11-5</td>
<td></td>
</tr>
<tr>
<td>EXP_FLOW-IND, 4-5</td>
<td></td>
</tr>
<tr>
<td>Expeditied Flow Indicator</td>
<td></td>
</tr>
<tr>
<td>decode, 4-5</td>
<td></td>
</tr>
<tr>
<td>encode, 11-4</td>
<td></td>
</tr>
<tr>
<td>Explicit Route</td>
<td></td>
</tr>
<tr>
<td>initial number, 4-7</td>
<td></td>
</tr>
<tr>
<td>number, 4-7</td>
<td></td>
</tr>
<tr>
<td>support indicator, 4-7</td>
<td></td>
</tr>
<tr>
<td>F, see FORWARD</td>
<td></td>
</tr>
<tr>
<td>FF, see SCRNFWD</td>
<td></td>
</tr>
<tr>
<td>FID, see Transmission Header</td>
<td></td>
</tr>
</tbody>
</table>
INDEX [continued]

FID-ID, 4-5
FID-LENGTH, 4-4
FID-POINTER, 4-4
FID0
decode, see Transmission Header
structure, A-10
use, A-9
FID1
decode, see Transmission Header
structure, A-11
use, A-9
FID2
building, 12-10, 12-11
DAF, 12-6, 12-9
decode, see Transmission Header
echoing SNF, 12-9
OAF, 12-8, 12-9
structure, A-12
use, A-9
FID3
local address field, 4-6
local system id, 4-6
structure, A-13
use, A-9
FID4
destination element field, 4-9
destination sub-area address, 4-9
explicit route number, 4-7
explicit/virtual route support indicator, 4-7
initial explicit route number, 4-7
network priority indicator, 4-7
origin element field, 4-9
origin sub-area address, 4-9
SNA indicator, 4-9
structure, bytes 0-7, A-14
structure, bytes 16-25, A-16
structure, bytes 8-15, A-15
transmission group information, 4-6
transmission priority, 4-7
use, A-10
virtual route information, 4-7, 4-8
FIDF
command format field, 4-9
command sequence number, 4-1
command type, 4-9
structure, A-16
use, A-10
FILE-> BUFFER, 13-19
Filename, recording, 3-3
FILL.BUFFER, 13-19
Filters, 8-1 to 8-15
activate/deactivate, 8-2, 8-3
frame layer, 8-9, 8-10
lead changes, 8-4, 8-5
request/response header, 8-13
selective LU, 8-7, 8-8
selective PU, 8-5 to 8-7
sense data, 8-14, 8-15
trace statements, 8-3, 8-4
transmission header, 8-11, 8-12
FK_IND, 13-16
Flags, A-6
FORMAT-IND, 4-11
FORM_LINK, 12-3
FORWARD, 3-3
Frame Layer
configuration, 9-6, 9-7
control field, 4-3, 11-2, A-6 to A-9
decode, 4-3, 4-4, 11-2
display format, 7-4
encode, 11-2, 11-3
filters, 8-9, 8-10
functions, A-5 to A-9
length, 4-2, 11-2
module 8/128, 24-4, 4-3
size, 9-6
state machine, 9-6
structure, A-5
window size, 9-9
Frame(s)
abort, 12-17
address, 4-3, 11-2
BCN, 12-15
CFGR, 12-15
decode, see Frame Layer
DISC, 12-5, 12-14
DM, 12-15
FRMR, 12-15
identifier(s), 13-4
length, 4-2, 11-2
NR/NI(S), 4-4, 11-2
numbered information, 12-16
P/F bit, 4-4, 11-3, 12-6
RD, 12-6, 12-14
REJ, 12-17
RIM, 12-6, 12-14
RNR, 12-7, 12-16
RR, 12-7, 12-16
SIM, 12-5, 12-15
SNRM, 12-5, 12-14
TEST, 12-15
test manager events, 13-3 to 13-14
transmitting, 13-20
UA, 12-14
unnumbered information, 12-16
UP, 12-15
user-defined, 12-7
XID, 12-6
FRAME-ADDR, 4-3
FRAME-ID, 13-3
FRAME-MODULO, 4-3
FRAME?, 13-5
FRAME.REJECT, 12-2
FREEZE, 3-4, 10-4
FRM.CHAR, 7-4
FRM.HEX, 7-4
FRM.OFF, 7-4
FRM.ON, 7-4
FROM.CAPT, 3-2
FROM.DISK, 3-2
FULL, 7-7
Function Management, A-20, A-21
GET_ID, 12-6
HALT, 3-2, 10-3
I/F_IND, 13-16
ID, 11-2
ID-OUT, 11-2
Identifier
DFC request/response unit, 13-12
FMD request/response unit, 13-11
frame, 13-4
NC request/response unit, 13-13
request/response unit category, 13-8
SC request/response unit, 13-12
transmission header, 13-7
IDLE_FLAG, 9-5
IDLE MARK, 9-5
IERN, 4-7
IF=NRZ, 2-3, 9-4
IF=NRZI, 2-3, 9-5
IF=NRZI+CLK, 2-3, 9-5
IF=NRZ_EXT, 2-3, 9-4
IF=V11, 2-1, 9-2
IF=V28, 2-1, 9-2
IF=V35, 2-1, 9-2
IF=V36, 2-2, 9-2
Information Field
anchored comparison, 13-13
defining, 13-21
numbered information frame, 12-16
unanchored comparison, 13-14
unnumbered information frame, 12-16
INDEX [continued]

Information Frame(s)
queuing, 13-6
resetting queue, 13-18
transmitting, 12-7, 12-16
user-defined, 12-7, 12-16

INT_MODE, 12-4

Interface
bit rate, 2-3
clocking, 9-4
interframe fill, 9-5
lead transitions, 13-17
leads, 2-2, 9-3
to DCE/DTE, 4-2, 9-2
V.11/X.21, 2-1, 9-2
V.28/RS-232C, 2-1, 9-2, 13-17
V.35, 2-1, 9-2
V.36, 2-2, 9-2, 13-17
INTERFACE_SPEED, 2-3, 9-4
L_BACKUP, 13-6, 13-18
L_OE, 13-6, 13-18

JIS8, 7-4
KILL_LINK, 12-4
L2_FRAME_POINTER, 11-2
L2=MOD128, 2-4
L2=MOD8, 2-4
Layer 1, see Physical Layer
Layer 2, see Frame Layer
Lead Transition(s), see Control Lead
LEAD_NUMBER, 4-3
LINK, 12-2
LINK#, 11-2, 11-3
Link(s)
disconnecting, 12-4
forming, 12-3
initializing, 12-4
resetting, 12-7
selecting, 9-6, 11-2
supported number, 9-6
Live Data
capturing to RAM, 5-1, 5-2
description, 10-1, 10-2
monitor, 3-1, 3-2
simultaneous playback, 3-4, 10-4
LOAD_RETURN_STATE, 13-2
Local Address Field, 4-6
Local System ID, 4-6
LOCAL_ADDRESS, 4-6
LONG_INTERVAL, 7-10
LU
filters, 8-7
LU, filters, 8-8
LU-PUID, 4-6
LU-SSCP, 4-6
M_CONTROL, 4-3
M_NR, 4-4
M_NS, 4-4
M_PF, 4-4
Mapping Field
decode, 4-5
encode, 11-3
test manager event, 13-8
MAPPING_FIELD, 4-5
MATCH, 13-13
MAX_FRAME_LENGTH, 9-6
Modulo 8/128
decode, 4-3
setting, 2-4
MONITOR, 3-2
Monitor
architecture, 3-1 to 3-4
bit rate, 2-3
classification, 2-1 to 2-4
decode, 4-1 to 4-15
live data, 3-1, 3-2
playback, 3-2 to 3-4
test script(s), 14-1 to 14-9
Multipoint, A-3
configuration, A-3
decoding, 9-7
test script, 14-21 to 14-26
MULTI_PT, 9-7
NDM, 12-2
Network Priority Indicator, 4-7
NEW_STATE, 13-2
NEW_TM, 13-2
NO_LINK, 12-2
NO_POLL, 12-6
NR, 11-2
NRM, 12-2
NS, 11-2
NTWK_PRIORITY, 4-7
OEF, 4-9
OK, 12-7
ORIG_ADD_FIELD, 4-6
Origin Address Field
decode, 4-5, 11-3
encode, 11-4
filters, 8-7, 8-8
test manager event, 13-7
user-defined, 12-8
Origin Element Field, 4-9
Origin Sub-area Address, 4-9
OSAF, 4-9
OTHER_EVENT, 13-16
P_F.Bit, 11-3
PACING_IND, 4-12
PADDED_DATA_IND, 4-13
Path Control, A-9 to A-16
Physical Layer
configuration, 2-1, 9-2 to 9-5
decode, 4-2, 4-3
description, A-4
filters, 8-4, 8-6
reconfigure, 9-1
test manager actions, 13-17
test manager events, 13-3
Physical Unit, see PU
PLAYBACK, 3-3
Playback
capture RAM, 3-2
decoding, 4-4
control, 3-3, 3-4
disk recording, 3-2
description, 10-3
monitor, 3-2 to 3-4
simultaneous live data, 3-4, 10-4
Point-to-point, A-2
decoding, 4-4
emulation, 9-7
Port Timer, setting, 9-7
Poll Timer, setting, 9-7
POL_FINAL_BIT, 9-7
PORT, 3-2
PORT_ID, 4-2
PRIMARY, 9-1
Primary Station
disconnecting a link, 12-4, 12-5, 12-7
decoding, 4-4
description, 9-7
equipment state machine, 12-1 to 12-3
establishing a link, 12-3
decoding, 9-7
point-to-point emulation, 9-7
retries, 9-9
Printer Configuration, 5-4
INDEX [continued]

Printing
  capture RAM, 5-4
disk recording, 5-4
throughput graph, 7-10
PRINT_OFF, 5-4
PRINT_ON, 5-4
PRINT_TPR, 7-10
PT_TO_PT, 9-7
PU
de code, 4-3, 11-2
definition, A-6
filters, 8-5 to 8-7
setting, 9-9, 11-2
test manager event, 13-5
QEMPTY?, 13-6
Queued Response Indicator, 4-12, 11-6
QUEUED-RESP-IND, 4-12
QUIT_TRA, 5-2
R1=ALL, 8-4
R1=NONE, 8-4
R2+, see Filters, frame layer
R2-, see Filters, frame layer
R=ASCII, 7-4
R=EBCDIC, 7-4
R=HEX, 7-4
R=JIS8, 7-4
RAM_FILTER_OFF, 8-3
RAM_FILTER_ON, 8-2
RD, 12-6
RDAF, 11-3
REC-LENGTH, 4-2
REC-POINTER, 4-3
RECORD, 6-2
Recording
  captured data, 5-4
  filename, 3-3
  live data to disk, 6-1, 6-2
  overwrite, 6-1
  playback disk, 3-2, 3-3
  stop, 6-2
  suspend, 6-2
Remote Control, 1-1
Report Generator, see Display Format
  REP_CHAR, 7-3
  REP_COMP, 7-2
  REP_FILTER_OFF, 8-2
  REP_FILTER_ON, 8-2
  REP_HEX, 7-3
  REP_NONE, 7-3
  REP_OFF, 7-2
  REP_ON, 7-2
  REP_SHORT, 7-2
Request/Response Header
  begin bracket indicator, 4-12, 11-6
  begin chain indicator, 4-11, 11-5
  building, 12-11
  change direction indicator, 4-12, 11-6
code selection indicator, 4-12, 11-6
decode, 4-10 to 4-13
definite response indicator, 4-11, 11-5
display format, 7-5
enciphered data indicator, 4-13, 11-7
encode, 11-4 to 11-7
end bracket indicator, 4-12, 11-6
end chain indicator, 4-11, 11-5
exception response indicator, 4-11, 11-5
filters, 8-13
format indicator, 4-11, 11-5
length, 4-10
pacing indicator, 4-12, 11-6
padded data indicator, 4-13, 11-7
queued response indicator, 4-12, 11-6
request/response indicator, 4-10, 11-4
response type indicator, 4-11, 11-5
sense data indicator, 4-11, 11-5
structure, A-18, A-19
test manager event, 13-8
user-defined, 12-11
Request/Response Unit
  building, 12-12
category, 4-10, 11-4
category identifier, 13-8
decode, 4-14, 4-15
DFO identifier, 13-12
display format, 7-6
encode, 11-8
FMD identifier, 13-11
length, 4-15
NC/SC identifier, 13-12
test manager event, 13-9
user-defined, 12-12
RESET, 12-7
RESET_QUEUES, 13-18
Response Type Indicator, 4-11, 11-5
RESPONSE-TYPE, 4-11
Retries
  number, 9-9
timer, 9-9
RETURN_FIELDS, 12-9
RETURN_STATE, 13-2
RFID+, see Filters, transmission header
RFID-0, 8-11
RFID-, see Filters, transmission header
RFID-0, 8-11
RFID=ALL, 8-12
RFID=NONE, 8-12
RFRM+, see Filters, frame layer
RFRM+SNRM, 8-9
RFRM-, see Filters, frame layer
RFRM-SNRM, 8-9
RFRM=ALL, 8-9
RFRM=NONE, 8-9
RFRMA=ALL, 8-5
RH-ID, 4-10
RH-LENGTH, 4-10
RH-POINTER, 4-10
RH-REQ-RES, 4-10
RH_CHAR, 7-5
RH_HEX, 7-5
RH_OFF, 7-5
RH_ON, 7-5
RHL_REQ, 13-8
RHL_RES, 13-8
RIM, 12-6
ROAF, 11-3
RR+, see Filters, request/response header
RRH+FMD, 8-13
RRH-, see Filters, request/response header
RRH+FMD, 8-13
RRH=ALL, 8-13
RRH=NONE, 8-13
RSD+, see Filters, sense data
RSD+PER, 8-14
RSD-, see Filters, sense data
RSD-PER, 8-14
RSD=ALL, 8-14
RSD=NONE, 8-14
RSNF, 11-3
RTHAF=ALL, 8-7
RTRACE, 8-9
RU-LENGTH, 4-15
RU-POINTER, 4-14
RUN_LAYER2, 12-3
RUN_SEQ, 13-2
RU_CHAR, 7-6
RU_HEX, 7-6
RU_OFF, 7-6
RU_ON, 7-6
?RX_BLU, 13-4
?RX_FRAME, 13-5
?RX_RH, 13-9
?RX_RU, 13-9
S.BRACKET-IND, 11-6, 12-11
INDEX [continued]

S.CHAIN-IND, 11-5, 12-11
S.CHANGE-DIR-IND, 11-6, 12-11
S.CODE-SEL-IND, 11-6, 12-11
S.ENOPH-DATA-IND, 11-7, 12-11
S.EXP-FLOW-IND, 11-4, 12-10, 12-11
S.FID-IND, 11-3, 12-10, 12-11
S.FORMAT-IND, 11-5, 12-11
S.MAPPING-FIELD, 11-3, 12-10, 12-11
S.PACING-IND, 11-6, 12-11
S.PADDED-DATA-IND, 11-7, 12-11
S.QUEUE-RESP-IND, 11-6, 12-11
S.RESPONSE-TYPE, 11-5, 12-11
S.RH-ID, 11-4, 12-11
S.RH-REQ-RES, 11-4, 12-11
S.RU-ID, 11-8, 12-12
S.SD-CODE, 11-7
S.SD-ID, 11-7
S.SENSE-DATA-IND, 11-5, 11-7, 12-11, 12-12
S.SFMR, 12-15
S.SDF, 12-14
S.SL, 12-15
S.R, 12-14
S.REJ, 12-17
S.RIM, 12-14
S.RNR, 12-16
S.RR, 12-16
S.SIM, 12-15
S.SNRM, 12-14
S.TEST, 12-15
S.SUA, 12-14
S.SUI, 12-16
S.SUP, 12-15
Screen(s)
clearing, 7-7
scrolling, 3-3, 3-4
SCRN_BACK, 3-3
SCRN_FWD, 3-3
SD-IND, 4-14
SD-LENGTH, 4-13
SD-POINTER, 4-13
SDAF, 11-4, 12-6, 12-11
SDLC Layer, see Frame Layer
SD_CHAR, 7-6
SD_HEX, 7-6
SD_OFF, 7-6
SD_ON, 7-6
?SEARCHLI, 13-14
SECONDARY, 9-1
Secondary Station(s)
configuration, 9-8, 9-9
disconnecting a link, 12-7
evolution, 9-1
evolution state machine, 12-2, 12-3
initializing a link, 12-4
number of retries, 9-9
retry timer, 9-9
selecting, 9-6, 11-2
setting PU, 9-6, 11-2
supported number, 9-6, 11-2
terminating a link, 12-4
window size, 9-9
#SECS, 11-2
SEE_TRA, 5-3
SEND, 12-7
SEND_BTU, 12-7
SEND_BUFFER, 13-21
SEND_DISC, 12-2
send_info, 12-16
SEND_SNRM, 12-2
send_test, 12-15
send UI, 12-16
Sense Data
category, 4-14, 11-7
code, 11-7
decode, 4-13, 4-14
display format, 7-6
encode, 11-7, 11-8
filters, 8-14, 8-15
indicator, 4-11, 11-5
length, 4-13
modifier, 11-7
structure, A-20, A-21
SENSE-DATA-IND, 4-11
SEQ-NUM-FIELD, 4-5
Sequence Number Field
code, 4-5, 11-3
encode, 11-4
user-defined, 12-9
Sequence Numbers
command (FIDF), 4-10
N(R), 4-4, 11-2
N(S), 4-4, 11-2
SNF, 4-5, 11-3, 11-4, 12-9
TGSNF, 4-8
VRSNF, 4-9
SEQ(? SEQ), 13-2
SET_DAF, 12-6
SET_LENGTH, 12-11 to 12-13
SET_LINK_ID, 9-9
SET_OAF, 12-6
SET_SNF, 12-9
SHORT-INTERVAL, 7-10
SIM, 12-5
SIM/UUA_WAIT, 12-2
Simulation, 14-10 to 14-20
Size
buffer, 13-18
frame, 9-6
SNA
indicator, 4-9
layers, A-3, A-4
SNAI, 4-9
SNRM, 12-5
SNRM/UUA_WAIT, 12-2
SOAF, 11-4, 12-11
SPEED=, see Bit Rate
SSNF, 11-4, 12-11
SSNF1, 11-4, 12-11
START_TIME, 4-2
STARTUP, 9-1
State Machine
changing emulation state, 12-2
primary emulation, 12-1, 12-2
secondary emulation, 12-2
test manager, 13-1
STATE_INIT( )STATE_INIT, 13-1
STATE( )STATE, 13-1
STRING->BUFFER, 13-19
TCLR, 13-1
Test Manager
action definition, 13-1
actions, 13-16 to 13-21
event recognition, 13-2 to 13-16
initialize the, 13-1
sequences, 13-2
state initialiization, 13-1
state transition, 13-2
stopping the, 13-2
subroutines in, 13-2
using buffers, 13-18 to 13-21
Test Script(s), 14-1 to 14-27
emulation, 14-10 to 14-27
monitor, 14-1 to 14-9
multiple, 13-2
multipoint, 14-21 to 14-26
TG_NON-FIFO-IND, 4-8
TG_SNF, 4-8
TG_SWEEP, 4-6
THAF, 13-7
Throughput Graph
display, 7-9
long interval, 7-10
INDEX [continued]

Throughput Graph [continued]
  printing, 7-10
  short interval, 7-10
TH_CHAR, 7-5
TH_HEX, 7-5
TH_OFF, 7-5
TH_ON, 7-5
TIMEOUT, 13-14
TIMER, 13-15
Timer(s)
  poll timer, 9-7
  retry, 9-9
  test manager event, 13-14, 13-15
  wakeup, 13-15
TIMER-NUMBER, 13-15
TIMER_IND, 13-16
Timestamp
  decode, 4-2
  display, 7-3
TIME_DAY, 7-3
TIME_OFF, 7-3
TIME_ON, 7-3
  TITLE, 3-3
TM_STOP, 13-2
TOP, 3-3
TO_DCE_IF, 9-2
TO_OTE_IF, 9-2
TPF, 4-7
TPR_OFF, 7-9
TPR_ON, 7-9
Trace Statements
  display format, 7-3, 7-9
  filters, 8-3, 8-4
  TEST, 12-15
TRANSFER, 5-2
Transmission Group, sweep indicator, 4-6
Transmission Header
  data count field, 4-6
  decode, 4-4 to 4-10, 11-3
  destination address field, 4-5, 11-3, 11-4, 12-8, 13-7
  display format, 7-5
  encode, 11-3, 11-4
  expedited flow indicator, 4-5, 11-4
  FID3 decode, see FID3
  FID4 decode, see FID4
  FIF decode, see FIDF
  FIFO indicator, 4-8
  filters, 8-11, 8-12
  identifier, 4-5, 11-3, 13-7
  length, 4-4
  mapping field, 4-5, 11-3, 13-8
  origin address field, 4-5, 11-3, 11-4, 12-8, 13-7
  sequence number, 4-8
  sequence number field, 4-5, 11-3, 11-4
  test manager event, 13-7
  types, A-9
Transmission Priority, 4-7
Transmitting
  abort, 12-17
  BCN, 12-15
  buffers, 13-20
  CFGR, 12-15
  CRC error, 12-17
  DISC, 12-6, 12-14
  DM, 12-15
  FRMR, 12-15
  information frame, 12-16
  RD, 12-6, 12-14
  REJ, 12-17
  RIM, 12-6, 12-14
  RNR, 12-7, 12-16
  RR, 12-7, 12-16
  SIM, 12-5, 12-15
  SNRM, 12-5, 12-14
  TEST, 12-15
  UA, 12-14
  UP, 12-15
  user-defined BTU, 12-7
  user-defined frame, 12-7
  XID, 12-6
  TRA_ALL, 6-3
  TRA_END, 5-3
  TRA_START, 5-3
  TWA, 9-7
  Two-Way Mode
    alternate, 9-7
    simultaneous, 9-7
  TWS, 9-7
  UI_QUEUE, 13-6, 13-18
  Virtual Route
    change window indicator, 4-8
    change window reply indicator, 4-9
    number, 4-7
    pacing count, 4-7
    pacing request, 4-8
    pacing response, 4-8
    reset window reply indicator, 4-9
    send sequence number, 4-9
    sequence & type indicator, 4-8
    support indicator, 4-7
    VR-CWI, 4-8
    VR-CWRI, 4-9
    VR-PAC-CNT-IND, 4-7
    VR-PQ, 4-8
    VR-PRS, 4-8
    VR-RWI, 4-9
    VR-SNF-SEND, 4-9
    VR-SQTI, 4-8
    VRN, 4-7
  7WAKEUP, 13-15
  WAKEUP_CPU, 2-1
  Wildcard(s)
    comparison, 13-13
    test manager events, 13-16
  Window, size, 9-9
  XID, 12-6

IDACOM
SDLC/SNA Programmer's Manual