
SS#7 PROGRAMMER'S MANUAL

November 1990
Version 2.0

PREFACE

This manual is intended to provide a programmer's guide to the SS#7 Monitor/Simulation programs. General programming information is provided in the Programmer's Reference Manual. Information contained in this manual is machine independent. Refer to the appropriate Protocol Set Reference Manual for a list of valid tokens which can be used in test scripts.

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 SS#7 Monitor and Emulation User Manuals for instructions to load and operate the software. 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.

Copyright © **Hewlett-Packard Company** 1989, 1990

P/N IDAC-601238

IDACOM
A division of Hewlett-Packard

4211 - 95 Street
Edmonton, Alberta
Canada T6E 5R6
Phone: (403) 462-4545
Fax: (403) 462-4869

TABLE OF CONTENTS

PREFACE

1	INTRODUCTION	1-1
2	MONITOR ARCHITECTURE	2-1
2.1	Live Data	2-1
2.2	Playback	2-2
	Playback Control	2-3
2.3	Simultaneous Live Data and Playback	2-4
3	MONITOR CONFIGURATION	3-1
3.1	Level 1	3-1
3.2	Level 2	3-3
3.3	Configuration Storage	3-4
3.4	Protocol Set Selection	3-5
4	CAPTURE RAM	4-1
4.1	Capturing to RAM	4-1
4.2	Transferring from RAM	4-2
	To Disk	4-3
	To Printer	4-4
5	DISK RECORDING	5-1
6	DISPLAY FORMAT	6-1
7	MESSAGE DECODE	7-1
7.1	Tokens	7-2
	Constructed Tokens	7-5
8	ROUTING LABELS	8-1

TABLE OF CONTENTS [continued]

9	FILTERS	9-1
9.1	Filter Activation	9-1
9.2	Filter Conditions	9-2
10	TRIGGERS	10-1
10.1	Trigger Control	10-1
10.2	Trigger Conditions	10-2
	Message and Parameter Triggers	10-3
	Other Triggers	10-6
10.3	Trigger Actions	10-6
10.4	Call Tracing	10-7
11	LEVEL 1 DECODER	11-1
12	SIMULATION ARCHITECTURE	12-1
12.1	Live Data	12-1
12.2	Playback	12-2
12.3	Simultaneous Live Data and Playback	12-3
13	SIMULATION CONFIGURATION	13-1
13.1	Level 1	13-1
13.2	Level 2	13-4
13.3	Configuration Storage	13-8
13.4	Protocol Set Selection	13-8
14	LEVEL 2 SIMULATION	14-1
14.1	Sequence Numbering	14-1
14.2	Send Commands	14-1
15	TEST MANAGER	15-1
15.1	ITL Constructs	15-1

TABLE OF CONTENTS [continued]

15 TEST MANAGER [continued]

15.2	Event Recognition	15-3
	Level 1	15-3
	Level 2	15-4
	Other Events	15-6

16 TEST SCRIPTS 16-1

16.1	LINK_UP.F	16-1
16.2	COO_CCITT.F	16-6

APPENDICES**A CODING CONVENTIONS A-1**

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

B ASCII/EBCDIC/HEX CONVERSION TABLE B-1**C COMMAND CROSS REFERENCE LIST C-1****D SAMPLE CONFIGURATION FILE D-1****INDEX**

LIST OF FIGURES

1-1	Sample Stack Comment	1-1
2-1	SS#7 Monitor Data Flow Diagram - Live Data	2-1
2-2	SS#7 Monitor Data Flow Diagram - Offline Processing	2-2
2-3	SS#7 Monitor Data Flow Diagram - Freeze Mode	2-4
3-1	Level 1 Monitor Configuration Menu	3-1
3-2	Level 2 Monitor Configuration Menu	3-3
4-1	SS#7 Data Flow Diagram - Capture to RAM	4-1
5-1	SS#7 Data Flow Diagram - Recording to Disk	5-1
6-1	Display Format Menu	6-1
12-1	SS#7 Simulation Data Flow Diagram - Live Data	12-1
12-2	SS#7 Simulation Data Flow Diagram - Offline Processing	12-2
12-3	SS#7 Simulation Data Flow Diagram - Freeze Mode	12-3
13-1	Level 1 Simulation Configuration Menu	13-1
13-2	Level 2 Simulation Configuration Menu	13-4

LIST OF TABLES

6-1	Functional Part Tokens	6-2
6-2	Detail Formats and Tokens	6-3
6-3	Dual Window Commands	6-4
7-1	Mapping Port Identifiers to Application Processors	7-1
13-1	Simulation Timers	13-7
A-1	ITL Symbols	A-2

1

INTRODUCTION

The SS#7 Monitor/Simulation is implemented as a layered set of protocol specifications. A partial or complete protocol set can be assembled by loading the appropriate national or international specification.

The SS#7 Monitor is not a state-driven monitor, i.e. it does not have knowledge of expected events. Rather, the monitor decodes the data and reports information on received events. Filters, triggers, RAM capture, and disk recording are also available.

The SS#7 Simulation provides additional tools necessary to build and transmit SS#7 messages on a transmit channel. All monitor functions are available for the receive channel.

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.

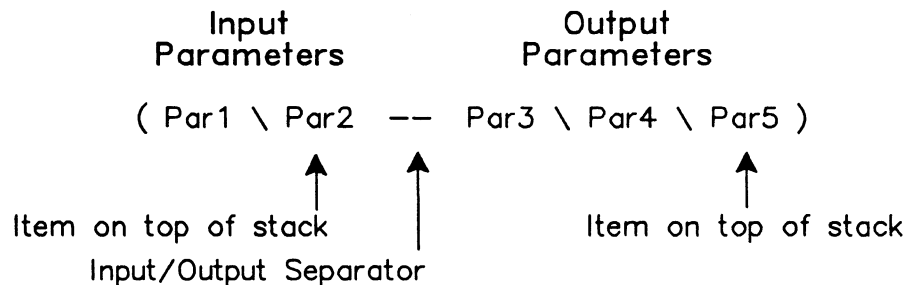


Figure 1-1 Sample Stack Comment



NOTE

See Appendix B for further explanation of stack parameters.

2

MONITOR ARCHITECTURE

The SS#7 Monitor program monitors live data, saves data to capture RAM or disk, and displays data in a number of different formats. Data can be passed through filters which limit the displayed, captured, recorded, or test manager data. Triggers perform specific actions when a specified event occurs.

2.1 Live Data

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

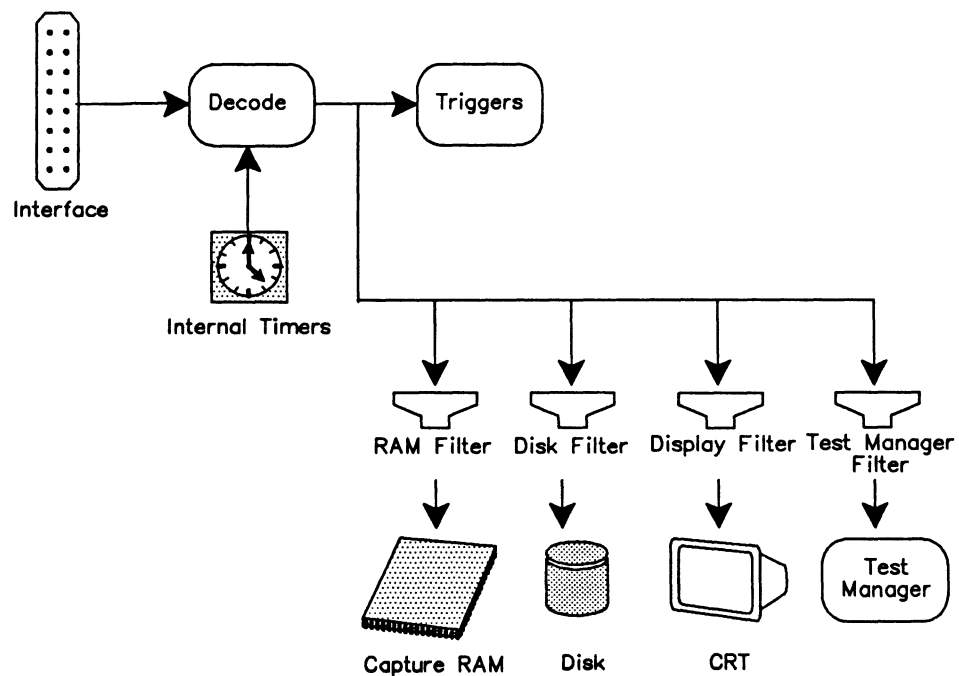


Figure 2-1 SS#7 Monitor Data Flow Diagram – Live Data

By default, the SS#7 Monitor simultaneously captures data in the capture RAM buffer and displays it on the screen in a short format report.

 **Display topic**
Live Data function key

MONITOR (--)

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

2.2 Playback

Data can be examined in an offline mode using either the capture RAM or the disk file as the data source.

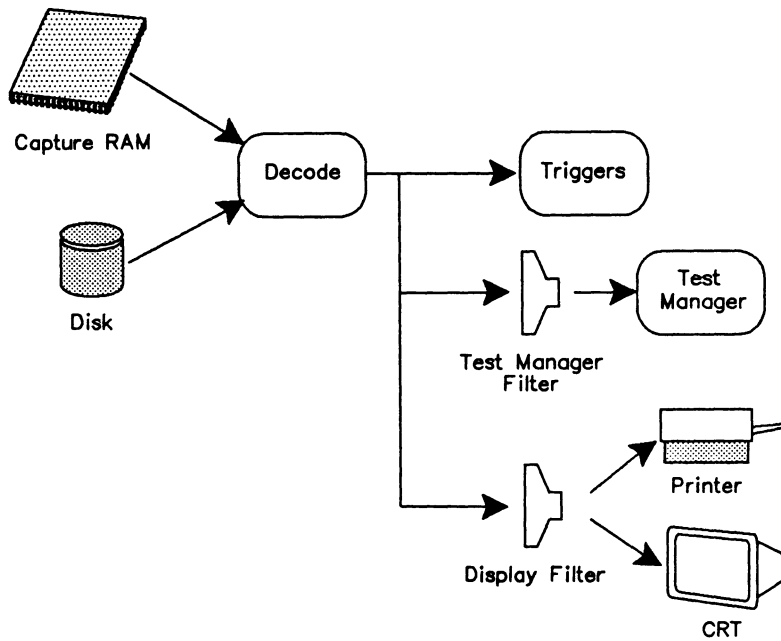



Figure 2-2 SS#7 Monitor Data Flow Diagram – Offline Processing

 FROM_CAPT HALT
Display topic
Playback RAM function key

 FROM_DISK HALT PLAYBACK
Display topic
Playback Disk function key

HALT (--)

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

```
FROM_DISK          ( Identify a disk file as data source )
HALT               ( Place the monitor in playback mode )
" DATA3" =TITLE   ( Create title for next data file to be opened )
PLAYBACK          ( Playback data )
```

Playback Control

The following commands can be used to control display scrolling.

FORWARD or F (--)

Scrolls one line forward on the screen.



↓ (Down arrow)

BACKWARD or B (--)

Scrolls one line backward on the screen.



↑ (Up arrow)

SCRN_FWD or FF (--)

Scrolls one page forward on the screen.



CTRL ↓

SCRN_BACK or BB (--)

Scrolls one page backward on the screen.



CTRL ↑

TOP (--)

Positions the display at the start of the playback source.



CTRL SHIFT ↑

BOTTOM (--)

Positions the display at the end of the playback source.

 CTRL SHIFT ↓

2.3 Simultaneous Live Data and Playback

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

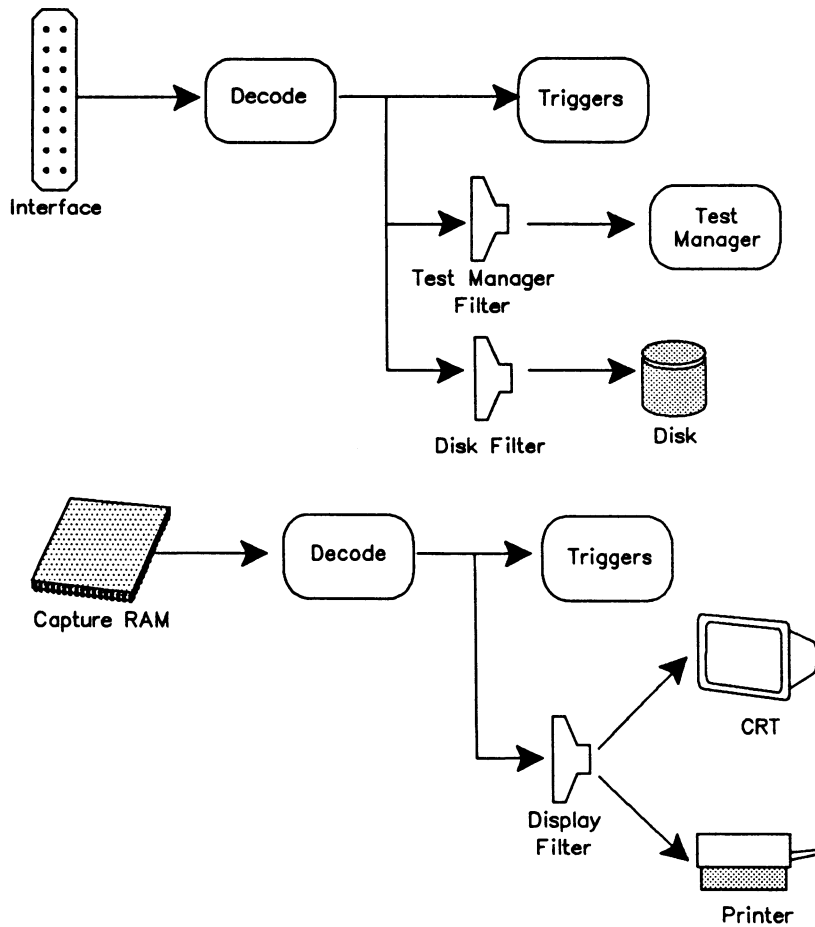



Figure 2-3 SS#7 Monitor Data Flow Diagram – Freeze Mode

 RECORD FROM_CAPTURE 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.

3**MONITOR CONFIGURATION**

This section describes the commands associated with each item on the Level 1 and Level 2 Configuration Menus and the Protocol Set Selection Menu.

3.1 Level 1

This section describes the commands to configure the physical interface and route data to the appropriate application processor or output device for WAN (wide area network) or PRA (primary rate access).

The Primary Rate interface is configured on the Home processor prior to loading the application.

The WAN interface is configured on the application processor after loading and switching to the SS#7 Monitor.

Level 1 Configuration Menu	
Signalling Data Link Level	
→ Interface Type	RS232C/V.28
Bit Rate	64000
BOF Timestamp	OFF


Figure 3-1 Level 1 Monitor Configuration Menu

Signalling Data Link Level

→ *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 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 connector and electrically isolates the other connectors on the port.

 RS449/V.36 function key

→ *Bit Rate* (WAN and PRA Interface)

The interface speed is measured, in bits per second, directly from the physical line.

SPEED@ (-- speed)

Returns the current interface speed.

 *Measure Speed* function key

→ *BOF Timestamp* (WAN and PRA Interface)

BOF-TSTAMP! (flag --)


Specifies whether beginning of frame timestamps are recorded on incoming messages.

 ON/OFF function key

The following commands control the flow of data to the application program.

OFF-LINE (--)

Turns the receivers off.

 **Monitor topic**
Online function key (not highlighted)

ON-LINE (--)

Turns the receivers on.

 **Monitor topic**
Online function key (highlighted)

3.2 Level 2

This section describes the commands and variables to configure level 2 of the SS#7 Monitor.

Level 2 Configuration Menu				
		Signalling Link Level		
→ SU Compression		MAX		
SUERM Function	ON	Octet Counting Mode	AUTO	
T Threshold	64	Maximum SIF	272	

Figure 3-2 Level 2 Monitor Configuration Menu

Signalling Link Level

→ SU Compression

COMPRESS-SU! (value --)

Sets the compression ratio of identical FISU's or LSSU's on live data to a specified value.

The SU_MAX value sets the maximum compression at 99,999,999 (default). The OFF value sets the compression to off.

 *Modify Count/MAX Compression/OFF function key*

COMPRESS-SU@ (-- value)

Returns the current SU compression ratio.

→ SUERM Function

SUERM-FUNCTION! (flag --)

Selects whether the SUERM (signal unit error rate monitor) will be active (default is on). When turned on, the error counter is reset to zero.

 *ON/OFF function key*

SUERM-FUNCTION@ (-- flag)

Returns true if the SUERM is active.

→ T Threshold

SUERM-THRESH! (value --)

Sets the SUERM threshold for determining a link failure indication. Valid values are 1 through 65535 (default is 64).

 *Modify Value function key*

SUERM-THRESH@ (-- value)

Returns the current value of the SUERM threshold.

SUERM-COUNT! (value --)

Sets the SUERM error counter.

SUERM-COUNT@ (-- value)

Returns the current value of the SUERM error counter.

→ *Octet Counting Mode*

OCTET-MODE! (mode --)

Selects the octet counting mode used by the monitor to ON, OFF, or AUTO (default).

 *ON/OFF/AUTO* function key

OCTET-MODE@ (-- mode)

Returns the current state of the octet counting function.

→ *Maximum SIF*

SIF-MAXSIZE! (value --)

Specifies the maximum SIF (signalling information field) size to turn octet counting off when the octet counting mode is set to AUTO. Valid values are 1 through 999 (default is 272).

 *Modify SIF* function key

SIF-MAXSIZE@ (-- value)

Returns the current value of the maximum SIF size.

3.3 Configuration Storage

The current settings of the level 1 and level 2 monitor configurations can be saved to disk for future retrieval.

SAVE_CONFIG (filename --)

Saves the current configuration.

Example:

Save the current configurations in a file named 'Config' on floppy drive DR0.

```
" DR0:Config" SAVE_CONFIG
```

 **Monitor topic**
Save Config function key

LOAD_CONFIG (filename --)

Loads a previously saved configuration.

 **Monitor topic**
Load Config function key

3.4 Protocol Set Selection

The decoder can be configured to use a particular set of protocol files. Refer to the appropriate Protocol Set Reference Manual for valid filenames.

LOAD_PROTOCOL_SET (filenames\number --)

Loads the specified protocol file(s). In addition, playback from a disk file is stopped and the display format, trigger, filter, and routing label settings are restored to their default configurations.

Example:

Load the CCITT TUP protocol with supporting lower layers.

```
" CCITT_TUP88.T" "CCITT_NET88.T" "CCITT_LINK88.T" 3 LOAD_PROTOCOL_SET
```



Monitor topic

Protocol Set Selection Menu

→ CCITT_TUP88

Select function key

→ CCITT_NET88

Select function key

→ CCITT_LINK88

Select function key

Load Protocols function key



NOTE

LOAD_PROTOCOL_SET should be used before loading test scripts. When a protocol set is loaded, test scripts are cleared.



WARNING

Colon definitions which compile message tokens will crash if the protocol set is changed after creating the colon definition.

SELECT_VAR (protocol set --)

Selects the specified protocol set.



Protocol Variance Menu

Select function key

LOAD_ALL (--)

Loads all functional parts of the protocol set selected with SELECT_VAR.



Protocol Set Selection Menu

All function key

Load function key

Example:

Load all available CCITT 1988 functional parts.

```
" CCITT_88" SELECT_VAR      ( Selects the CCITT 1988 protocol set )
LOAD_ALL                    ( Loads all functional parts )
```


4

CAPTURE RAM

This section describes the data flow diagram for capture to RAM and lists the commands available for test scripts. Data stored in either capture RAM or disk can be played back as described in Section 2.2. Data stored in capture RAM can be transferred to disk.

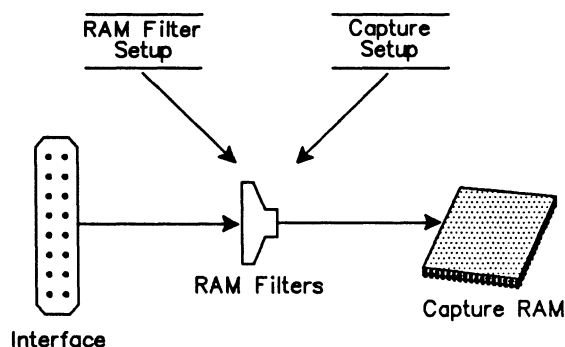


Figure 4-1 SS#7 Data Flow Diagram – Capture to RAM

4.1 Capturing to RAM

CAPT_ON (--)

Saves live data in capture RAM (default).

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


CAPT_OFF (--)

Live data is not saved in capture RAM.

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


CAPT_WRAP (--)

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

 **Capture** topic
Recording Menu
→ When Buffer Full
WRAP function key

CAPT_FULL (--)

Initializes capture RAM so that capturing stops when the buffer is full.

 **Capture topic**
Recording Menu
→ *When Buffer Full*
STOP function key

WARNING

CAPT_WRAP and *CAPT_FULL* erase all data in capture RAM.

CLEAR_CAPT (--)

Erases all data currently in capture RAM.

 **Capture topic**
Clear function key

4.2 Transferring from RAM

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

TRANSFER (--)

Transfers data from the selected data source.


 **Capture topic**
Save RAM to Disk function key

QUIT_TRA (--)

Abruptly terminates the transfer of data from the selected data source.


TRA_ALL (--)

Transfers the entire contents of capture RAM (default) or disk 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 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 port identifier and block number for the initial and final blocks selected for transfer in the Command and Test Script Windows.

Example:

Open a data file with the filename 'DATA3' and transfer all data from capture RAM to disk. After the transfer is complete, turn off data recording.

```

FROM_CAPT           ( Designate capture RAM as data source )
HALT                ( Enter playback mode )
" DATA3" =TITLE    ( Assign filename DATA3 )
RECORD              ( Open data recording )
TRA_ALL             ( Select all blocks for transfer )
CTOD_ON             ( Enable capture transfer to disk )
TRANSFER            ( Transfer data from capture to disk )
DISK_OFF            ( Turn off data recording )

```

To Disk
CTOD_ON (--)

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

CTOD_OFF (--)

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

To Printer

PRINT_ON (--)

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



Print topic

Print On function key

PRINT_OFF (--)

Data is not printed during playback (default).



Print topic

Print Off function key

Example:

Transfer all data from capture RAM to the printer.

```
FROM_CAPT      ( Designate capture RAM as data source )
HALT           ( Enter playback mode )
PRINT_ON       ( Enable printing )
TRA_ALL        ( Transfer all )
TRANSFER       ( Transfer data to printer )
```

5

DISK RECORDING

Live data from the interface can be recorded to either a floppy or hard disk. Data stored in either capture RAM or disk can be played back as described in Section 2.2. Data stored in capture RAM can be transferred to disk as described in Section 4.2.

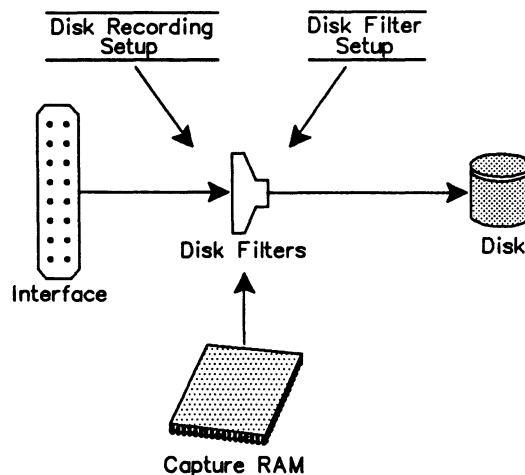



Figure 5-1 SS#7 Data Flow Diagram – Recording to Disk


DISK_WRAP (--)

Selects disk recording overwrite.

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

DISK_FULL (--)

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

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

WARNING

DISK_WRAP and DISK_FULL must be called prior to opening a recording with the RECORD command. If called while recording is in process, the status of the disk recording overwrite for this recording session will not change.

RECORD (--)

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

Example:

```
RECORD DATA1
```



Capture topic

Record to Disk function key (highlighted)



NOTE

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

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

Example:

```
Recording Start : SS#7 Mon           WAN Port 1 RS232-C  
V2.0-2.0             PT500 - 19      SN# 01-261
```

```
Recording End   : SS#7 Mon           Wan Port 1 RS232-C  
V2.0-2.0             PT500 - 19      SN# 01-261
```

DISK_OFF (--)

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



Capture topic

Record to Disk function key (not highlighted)

DIS_REC (--)

Momentarily suspends disk recording. The data recording file remains open but no data is saved to disk.



Capture topic

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

ENB_REC (--)

Enables data recording. The data recording file remains open and live data is recorded to disk.



Capture topic

Record to Disk function key (highlighted)
Suspend Recording function key (not highlighted)

Data from two or more channels can be recorded to the same file. The data file must be opened from each application and the filename, as specified by =TITLE, must be the same for each application. When the RECORD command is issued, data from that application's channel is included in the file.

The DISK_OFF command must be executed from each application to end that data recording. The file is closed when the last DISK_OFF command is executed.

6

DISPLAY FORMAT

The SS#7 Monitor and Simulation applications can display live data or data played back from capture RAM or a disk recording in a variety of formats. This section describes the commands to select the display format.

Display Format Menu			
→ Display Format	COMPLETE	Dual Window	OFF
Header		Trace Display Format	SHORT
Link	MNEMONIC		
Network	MNEMONIC	Timestamp	OFF
Detail			
Network	COMPLETE	Character Set	---
SCCP	COMPLETE		
TUP	COMPLETE	Throughput Graph	OFF
ISUP	COMPLETE	Short Interval (sec)	10
TCAP	COMPLETE	Long Interval (sec)	600
Other	HEX	Maximum Scale (%)	100

Figure 6-1 Display Format Menu

→ *Display Format*

The default display is short format. Header and detail control formats can only be modified when *Display Format* is set to *COMPLETE*.

REP_ON (--)

Turns on data display (default).

 OFF function key (not highlighted)


REP_OFF (--)

Turns off data display.

 OFF function key (highlighted)

REP_SHORT (--)

Displays data in a condensed report (default).

 SHORT function key

REP_COMP (--)

Displays data in a comprehensive report.

 *COMPLETE* function key

 **NOTE**

In complete format, protocol specific formats can be modified with the `HEADER_FORMAT` and `DETAIL_FORMAT` commands.

REP_CHAR (--)

Displays data in the currently selected character set.

 *CHAR* function key

REP_HEX

Displays data in hexadecimal format.

 *HEX* function key


REP_TRACE (--)

Displays only trace statements.

 *TRACE* function key

REP_SPLIT (--)

Displays data in short format with a split screen display.

 *SPLIT* function key

Tokens referring to entire functional parts (eg. <SU>, <MSU>, etc.) can be used with the `HEADER_FORMAT` and `DETAIL_FORMAT` commands to change display formats. The following table shows the relationship between these commands and each functional part.

Protocol Level	Token	HEADER_FORMAT	DETAIL_FORMAT
Link	<SU>	X	
Network	<MSU>	X	X
SCCP	<SCCP>		X
TUP	<TUP>		X
ISUP	<ISUP>		X
TCAP/OMAP	<TCAP>		X

Table 6-1 Functional Part Tokens

Header

Selects the format for the first display line.

- *Link*
- *Network*

HEADER_FORMAT (token \ format --)

Selects the header format for the specified protocol level.

Valid tokens are <SU> for link level, or <MSU> for network level. The format can be any of the following: #OFF, #HEX, #MNEM, or #TEXT.

 OFF/HEX/MNEMONIC/TEXT function keys

Detail

Selects the format for the second and subsequent display lines.

- *Network*
- *SCCP*
- *TUP*
- *ISUP*
- *TCAP*

DETAIL_FORMAT (token \ format --)

Selects the detail format for the specified protocol level. Valid formats and tokens are listed in the following table.

Protocol Level	Token	Format					
		#OFF	#HEX	#CHAR	#MSG	#PARS	#COMP
Network	<MSU>	✓	✓	✓			✓
SCCP	<SCCP>	✓	✓	✓	✓	✓	✓
ISUP	<ISUP>	✓	✓	✓	✓	✓	✓
TUP	<TUP>	✓	✓	✓	✓		✓
TCAP	<TCAP>	✓	✓	✓	✓	✓	✓

Table 6-2 Detail Formats and Tokens

- *Other*

Selects the format for undecoded protocol information.

OTHER_FORMAT (format --)

Selects the display format for undecoded protocol information. Valid values are #OFF, #HEX, or #CHAR.

 OFF/HEX/CHAR function keys

→ *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 (default).

Dual window commands vary depending on the machine configuration. Table 6-3 shows the relationship between machine configuration, application processors, and dual window commands.

Machine Type	Command	Dual Window AP #	
WAN/WAN	DUAL_1+2	AP #1	AP #2
BRA/WAN	DUAL_1+2	AP #1	AP #2
	DUAL_1+7	AP #1	AP #3
	DUAL_2+7	AP #2	AP #3
PRA	DUAL_3+4	AP #1	AP #2
PRA/BRA/WAN	DUAL_1+2	AP #1	AP #2
	DUAL_1+3	AP #1	AP #4
	DUAL_1+4	AP #1	AP #5
	DUAL_1+7	AP #1	AP #3
	DUAL_2+3	AP #2	AP #4
	DUAL_2+4	AP #2	AP #5
	DUAL_2+7	AP #2	AP #3
	DUAL_3+4	AP #4	AP #5
	DUAL_3+7	AP #4	AP #3
	DUAL_4+7	AP #5	AP #3
BRA/BRA	DUAL_1+2	AP #1	AP #2
	DUAL_1+3	AP #1	AP #4
	DUAL_1+4	AP #1	AP #5
	DUAL_1+5	AP #1	AP #6
	DUAL_1+7	AP #1	AP #3
	DUAL_2+3	AP #2	AP #4
	DUAL_2+4	AP #2	AP #5
	DUAL_2+5	AP #2	AP #6
	DUAL_2+7	AP #2	AP #3
	DUAL_3+4	AP #4	AP #5
	DUAL_3+5	AP #4	AP #6
	DUAL_3+7	AP #4	AP #3
	DUAL_4+5	AP #5	AP #6
	DUAL_4+7	AP #5	AP #3
DUAL_5+7	AP #6	AP #3	
PRA/WAN	DUAL_1+3	AP #1	AP #2
	DUAL_1+4	AP #1	AP #3
	DUAL_3+4	AP #2	AP #3

Table 6-3 Dual Window Commands

→ *Trace Display Format*

Selects the display format for trace statements.

TRACE_SHORT (--)

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

 *SHORT* function key

TRACE_COMP (--)

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

 *COMPLETE* function key

→ *Timestamp*

Timestamp reporting is available when the display format is not in split mode.

TIME_ON (--)

Displays the start and end timestamps as minutes, seconds, and tenths of milliseconds.

 *MM:SS.ssss* function key

TIME_OFF (--)

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

 *OFF* function key

TIME_DAY (--)

Displays the start and end timestamps as days, hours, minutes, and seconds.

 *DD HH:MM:SS* function key

→ *Character Set*

Selects the character set for data display.

CS=ASCII (--)

Sets the character set for data display to ASCII.

 *ASCII* function key

CS=EBCDIC (--)

Sets the character set for data display to EBCDIC.

 *EBCDIC* function key

CS=HEX (--)

Sets the character set for data display to hex (default). Each character is composed of two hexadecimal digits indicating the byte value of the character.

 *HEX* function key

CS=JIS8 (--)

Sets the character set for data display to JIS8.

 *JIS8* function key

→ *Throughput Graph*

The throughput rate can be calculated, displayed as a bar graph, and printed out. The SS#7 Monitor calculates throughput by totalling MSU bytes received, including one flag per MSU.

 **NOTE**

For accurate throughput measurement, the bit rate (line speed) must be set to match the actual line speed.

TPR_ON (--)

Calculates and displays the throughput rate as a bar graph.

 *DISPLAY* function key

TPR_OFF (--)

The throughput rate is not calculated or displayed (default).

 *OFF* function key

PRINT_TPR (--)

Calculates and displays the throughput rate as a bar graph, and prints the long term interval measurements.

 *DISPLAY AND PRINT* function key

→ *Short Interval (sec)*

Sets the short time interval, in seconds, for measuring, displaying, and printing the throughput results.

SHORT-INTERVAL! (value --)

Changes the current duration of the short interval (default is 10 seconds).

 *Modify Short Interval* function key

→ *Long Interval (sec)*

Sets the long time interval, in seconds, for measuring, displaying, and printing the throughput results.

LONG-INTERVAL! (value --)

Changes the current duration of the long interval (default is 600 seconds).


 *Modify Long Interval* function key

→ Maximum Scale (%)

Specifies the display size (percentage) of the throughput graph.

SCALE! (value --)

Modifies the current value of the maximum scale (default is 100 percent).

 **Modify Maximum Scale** function key

The following commands are used to select the display format for point codes and signalling link selection information.

RL_TEXT (--)

Displays point codes in decimal format (default).

 **Labels topic**
DECIMAL function key

RL_HEX (--)

Displays point codes in hexadecimal format.

 **Labels topic**
HEX function key

7

MESSAGE DECODE

The message decoder disassembles signal units into parts. The following variables are set each time a signal unit (frame) is received.

REC-POINTER (-- address)

Contains the memory address of the first byte of the received SU (i.e. backward sequence/indicator byte).

Example:

Store the first two bytes of the received SU into a variable.

```
0 VARIABLE BSN-FSN
<FIB> ?RX_FRAME
IF
    REC-POINTER @ W@ BSN-FSN !
ENDIF
```

REC-LENGTH (-- address)

Contains the number of bytes in the received SU. This number does not include any flags or CRC bytes.

PORT-ID@ (-- direction\port)

Returns the direction and port identifier for live or recorded data. Direction is dependent on whether the application is in monitor or simulation mode:

Direction	Monitor	Simulation
0	data stream 1	received data
1	data stream 2	transmitted data

The port identifier is dependent on the machine configuration:

Port	D-Channel	WAN	PRA	PRA/WAN	BRA	BRA/WAN	BRA/BRA	PRA/BRA/ WAN	WAN/WAN
0	Main	Main	Main	Main	Main	Main	Main	Main	Main
1		AP#1		AP#1	AP#1	AP#1	AP#1	AP#1	AP#1
2					AP#2	AP#2	AP#2	AP#2	AP#2
3			AP#1	AP#2			AP#4	AP#4	
4			AP#2	AP#3			AP#5	AP#5	
5							AP#6		
7	AP#3				AP#3		AP#3	AP#3	

Table 7-1 Mapping Port Identifiers to Application Processors

BLOCK-COUNT (-- address)

Contains the block sequence number for the frame. Every received frame is assigned a unique sequence number. BLOCK-COUNT initially contains zero and is incremented by one each time a frame is received.

<COPIES>@ (-- value)

Returns the number of copies of identical FISU or LSSU frames received when SU compression is active. In this case, BLOCK-COUNT contains the first block sequence number of the compressed group. For uncompressed SU's the value returned is 1.

START-TIME (-- address)

Contains the 48 bit timestamp for the start of signal unit or compressed frame groups. START-TIME can be used with the GET_TSTAMP_MILLI or GET_TSTAMP_MICRO commands as described in the Programmer's Reference Manual.



NOTE

The start time is not valid unless BOF timestamps were on when receiving live data.

END-TIME (-- address)

Contains the 48 bit timestamp for the end of frame or compressed frame group.

7.1 Tokens

The SS#7 Protocol contains unique words which can be used in ITL commands to reference protocol related information. Each of these words return a single unique value and are referred to as tokens. Tokens are identifiers for the basic lexical units of a protocol.

The SS#7 tokens identify the following structures in the protocol:

- The entire signal unit
- Each protocol level or functional part
- Messages within each functional part
- Parameters within each message
- Fields within each parameter
- Value instances within each field

Where appropriate, tokens are different for optional and mandatory parameters. Structures within the protocol are identified using more than one token when a single token is insufficient to perform the required functions.



NOTE

Although the TCAP nomenclature differs from other parts of SS#7, the concept of messages and parameters applies equally to TCAP transactions, components, and information elements.

Tokens are named using acronyms surrounded by angular brackets (eg. <CDPA>). Spare bits are often identified with a 0 appended to the associated token acronym (eg. <SLS0>).

Specific instances of tokens are identified by separating the field name and the instance name with an equal sign (eg. <SSN=ISUP> identifies the ISDN User Part instance of the Subsystem Number field).

Token qualifications are identified by separating the qualifier and the token acronym with a period (eg. <CDPA.SSN> identifies the Subsystem Number in the Called Party Address).

Specific instances and qualifications can also be combined (eg. <CDPA.SSN=ISUP> identifies the ISDN User Part instance of the Subsystem Number within the Called Party Address).

No token is named in instances where the protocol element would provide only a small benefit (eg. the token for the 1 bit odd/even indicator field in the SCCP Called Party Address is <CDPA.OE>. The addition of instance names for each value of this field are not included).

Tokens can be used in SS#7 test scripts to:

- change display formats for protocol related information;
- set and reset filter and trigger information;
- test for the presence and error status of fields within a received message;
- access field values within a received message; and
- access constants associated with fields within the protocol.

Refer to the appropriate Protocol Set Reference Manual for a list of usage symbols and valid tokens.

WARNING

Colon definitions which compile message tokens will crash if the protocol set is changed after creating the colon definition.

For decoding, tokens generally fall into one of two categories:

- Tokens refer to a protocol element with a value extracted by the message decoder. These tokens are identified with either the '?' or '@' usage symbols. When identified with '?' or '@', the <>L@ command returns the length (in bits) of the element. When identified with '?', the <>@ and <>@_ALL commands return the address of the extracted information's location. When identified with '@', the <>@ and <>@_ALL commands return the value of a 32 bit (or less) fixed length protocol element.
- Tokens refer to an instance of a protocol element. Tokens which are not identified with '?' or '@', can be used with the <># command to return the value of the instance. The '#' usage symbol marks an exception (i.e. the token can be used with commands from both categories).

<># (token -- value)

Returns a constant identifying the bit mask used by the decoder to detect a token (eg. the bit mask for <COA> is 0010 which corresponds to the decimal value 2).

Example:

Send an LSSU message of type SIO with no error.

```
CRC <SIO> <># SEND_LSSU
```

<>@ (token -- [value]\flag)

Returns a value and a true flag if a valid value is received for the specified token. The flag is used to ensure that the value is valid. A false flag is returned if the specified token is invalid or not present in the received SU. For variable length tokens, or tokens exceeding 32 bits, the value is the address where the token is located. In these cases, the command <>L@ can be used to determine the token length.

Example 1:

Define the actions for each CHM message. The <CHM> token is identified with the '@' usage symbol.

```
<CHM> <>@
IF
  DOCASE
    CASE <COO> <># [ action 1 ] ( action on receipt of changeover message )
    CASE <COA> <># [ action 2 ] ( action on receipt of changeover
                                acknowledgement )
    CASE <CBD> <># [ action 3 ] ( action on receipt of changeback )
    CASE <CBA> <># [ action 4 ] ( action on receipt of changeback
                                acknowledgement )
    CASE DUP      [ error action ]
  ENDCASE
ENDIF
```

Example 2:

Store the contents of the called party number address field into the DEST_STRING variable. <CDPN.ADDI> is the token for the address field and is identified with the '?' usage symbol.

```
<CDPN.ADDI> <>@
IF
  DEST_STRING
  <CDPN.ADDI> <>L@ 8 /
  CMOVE
ENDIF
```

<>@_ALL (token -- [value...\value]\count)

Returns all values of the specified token and a count of the total number found. If no tokens are found in the decoded frame, the count is 0. <>@_ALL is used where more than one token can be present in a message (eg. TCAP components).

<>L@ (token -- length)

Returns the length, in bits, of the specified token.

Constructed Tokens

When more than one copy of a token is present in a message, tokens can be constructed using the following set of commands.

`<: value token [OF value token] :> (-- token)`

Returns a constructed token.

Constructed tokens can be used when more than one copy of a token is present in a message. A constructed token identifies the n^{th} occurrence of the token, relative to the start of a message or an embedded constructed token. Constructed tokens can be embedded to any depth by separating each value/token pair with the OF command. The constructed token can then be used as input to the `<>@` and `<>L@` commands. If the n^{th} occurrence cannot be found, the result is 0.

Examples:

<code><: 2 <INKID> :></code>	<code>(Second occurrence of <INKID> invoke id)</code>
<code><: 1 <INKID> OF 1 <RER> :></code>	<code>(First occurrence of <INKID> following first)</code>
	<code>(Occurrence of <RER> - return error component)</code>

8

ROUTING LABELS

Up to ten routing labels can be configured for both triggers and filters. Routing labels can be edited, stored, and retrieved from disk.

->PC (msp\isp\lsp -- point_code)

Converts the subfields of a point code into a single point code. The subfields are ordered as most significant part to least significant part. The size of each subfield partition corresponds to the partition display option under the **Labels** topic.

PC-> (point_code -- msp\isp\lsp)

Converts a single point code into individual subfields.

SET_OPC (mask_subfields\point_code_subfields\label# --)

Sets the OPC (origination point code) of the specified label (1 through 10) according to the point code subfields, and sets the label's mask according to the mask subfields. The mask subfields are used to indicate which bits in the label are "don't care". Mask bits set to 1 imply "don't care".

Example (CCITT format):

Set the label 1 origination code for zone 2 (msp), network identifier 036 (isp), and a "don't care" signalling point identifier (lsp).

```
0 0 7 2 36 0 1 SET_OPC
```



Labels topic

Routing Label Selection Menu

→ *an OPC Value*

Set Point Code function key

SET_DPC (mask_subfields\point_code_subfields\label# --)

Sets the DPC (destination point code) of the specified label (1 through 10) according to the point code subfields, and sets the label's mask according to the mask subfields.



Labels topic

Routing Label Selection Menu

→ *a DPC Value*

Set Point Code function key

SET_SLS (mask_value\SLS_value\label# --)

Sets the SLS (signalling link selection) and the mask of the specified label (1 through 10).

Example:

Set the SLS of label 3 to binary value 01111 (in ANSI format) with the most significant bit set to "don't care".

```
0b10000 0b01111 3 SET_SLS
```



Labels topic

Routing Label Selection Menu

→ *an SLS Value*

Set SLS function key



NOTE

The menu limits "don't care" values to a single field whereas the previous commands allow "don't care" specifications on each bit. If any bits of a field are set to "don't care", the entire field will be displayed with x's.

SAVE_LABELS (filename --)

Saves the current set of labels into the specified file.



Labels topic

Save Routing Labels function key

Example:

Save the current set of labels in a file named 'Labels1' on disk drive WD1.

```
" WD1:Labels1" SAVE_LABELS
```

LOAD_LABELS (filename --)

Loads the contents of the specified file into the current set of labels.



Labels topic

Load Routing Labels function key

RESET_ROUTING_LABELS (--)

Resets all routing label fields to 0.

9

FILTERS

Filters provide the capability of passing or blocking specific events from the display, capture RAM, disk recording, or test manager. These four filters act independently. This section describes the commands used to pass or block individual events, and activate or deactivate each of the four filters. When used with triggers, filters can be used to trace a call (see Section 10.4).

9.1 Filter Activation

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

ACTIVATE_REPORT (--)

Activates the display (report) filter.



Filters topic

Activate Display Filter function key (highlighted)

DEACTIVATE_REPORT (--)

Deactivates the display filter.



Filters topic

Activate Display Filter function key (not highlighted)

ACTIVATE_CAPTURE (--)

Activates the capture RAM filter.



Filters topic

Activate RAM Filter function key (highlighted)

DEACTIVATE_CAPTURE (--)

Deactivates the capture RAM filter.



Filters topic

Activate RAM Filter function key (not highlighted)

ACTIVATE_DISK (--)

Activates the disk filter.



Filters topic

Activate Disk Filter function key (highlighted)

DEACTIVATE_DISK (--)

Deactivates the disk filter.



Filters topic

Activate Disk Filter function key (not highlighted)

ACTIVATE_TEST (--)

Activates the test manager filter.



Filters topic

Activate Test Manager function key (highlighted)

DEACTIVATE_TEST (--)

Deactivates the test manager filter.



Filters topic

Activate Test Manager function key (not highlighted)

 **NOTE**

The test manager filters reduce the processing duration of test scripts.

9.2 Filter Conditions

Filter conditions are set using a common set of commands for display, capture RAM, disk, and test manager. A filter must be selected before the conditions are specified.

R_FILTER (--)

Selects the display (report) filter.

C_FILTER (--)

Selects the capture RAM filter.

D_FILTER (--)

Selects the disk filter.

T_FILTER (--)

Selects the test manager filter.

The following commands are used to pass/block messages and parameters in the currently selected filter by identifying specific tokens. Refer to the appropriate Protocol Set Reference Manual for a list of usage symbols and valid tokens.

Tokens identified with a 'V' usage symbol and:

- a '?' (eg. variable length fields) can be used with the F_STRING command.
- a '@' can be used with the F_VALUE command. Tokens which are used with fields formatted with point codes can also be used with the F_PC_VALUE command.

Tokens identified with a 'G' can be used with the F+ALL and F-ALL commands. Tokens identified with an 'F' can be used with the F+, F-, and F* commands.

**NOTE**

In some cases, not all three of the F+, F-, and F commands are relevant. In general, parameter fields can be used with all of these commands, whereas message tokens and parameter instance tokens cannot be used with F*.*

F+ (token --)

Passes the specified token.

F- (token --)

Blocks the specified token.

F* (token --)

Turns off the specified token.

F+ALL (token group --)

Passes all valid token values within the specified token group.

F-ALL (token group --)

Blocks all valid token values within the specified token group.

Example:

Set the display filter to block FISU's, all valid 1 byte LSSU's, and all SNMM's.

```
R_FILTER
<FISU> F-
<LSSU> F-ALL
<SNMM> F-
```

F_VALUE (mask\value\token --)

Sets the filter value for the specified token when the mask is 0; sets the filter value to "don't care" when the mask is -1.

Example:

Set the CIC (circuit identification code) filter value to "don't care".

```
-1 0 <CIC> F_VALUE
```

F_STRING (mask\string\token --)

Sets the filter string for the specified token when the mask is 0; sets the filter string to "don't care" when the mask is -1.

Example:

Set the digits filter in the SCCP called party address to 4624545.

```
0 X" 4624545" <CDPA.ADDI> F_STRING
```

F_PC_VALUE (mask sub fields\point code sub fields\token --)

Sets the filter value for point code formatted tokens. See Section 8 for point code formats.

Example:

Set the signalling point code filter in the SCCP called party address to 3-45-1.

```
0 0 0 3 45 1 <CDPA.SPC> F_PC_VALUE
```

F+ALL_OTHERS (--)

Passes all tokens in undecoded protocol levels.

F-ALL_OTHERS (--)

Blocks all tokens in undecoded protocol levels.

Complete levels of filtering can be reset to their initial state using the following commands. Tokens used to identify protocol levels are listed in Table 6-2.

F_RESET (token --)

Resets all filter items in the specified protocol level.

F_RESET_ALL (--)

Resets all filters.

TF_RESET (token --)

Resets all triggers and filters in the specified protocol level.

TF_RESET_ALL (--)

Resets all triggers and filters.



NOTE

TF_RESET and TF_RESET_ALL do not require a trigger of filter to be initially selected.

These reset commands do not affect routing labels. Refer to the RESET_ROUTING_LABELS command to reset labels to 0; and TURN_OFF_LABEL_FILTERS to turn off label filters.

Routing labels (1 through 10) can be filtered using the following commands.

F+PC (label# --)

Passes the specified routing label.

F-PC (label# --)

Blocks the specified routing label.

F*PC (label# --)

Turns off filtering (default) for the specified routing label.

Example:

For the display filter, block routing labels 2 and 3, and turn off filtering on routing label 7.

```
R_FILTER
2 F-PC
3 F-PC
7 F*PC
```

TURN_OFF_LABEL_FILTERS (--)

Turns off all routing label filters.

Example:

Turn off all routing label filters to the display.

```
R_FILTER
TURN_OFF_LABEL_FILTERS
```

Trace reports can be turned on (default) or off using the following commands.

RTRACE (flag --)

YES RTRACE turns the display trace reports on. NO RTRACE turns them off.

CTRACE (flag --)

YES CTRACE turns the RAM capture trace reports on. NO CTRACE turns them off.

DTRACE (flag --)

YES DTRACE turns the disk recording trace statements on. NO DTRACE turns them off.

10

TRIGGERS

Four independent triggers provide the capability of performing specific actions when a certain event occurs. This section describes the commands to set the events and actions for each trigger.

10.1 Trigger Control

TR1_ON (--)

Turns on trigger 1.



Triggers topic

Arm Trig #1 function key (highlighted)

TR2_ON (--)

Turns on trigger 2.



Triggers topic

Arm Trig #2 function key (highlighted)

TR3_ON (--)

Turns on trigger 3.



Triggers topic

Arm Trig #3 function key (highlighted)

TR4_ON (--)

Turns on trigger 4.



Triggers topic

Arm Trig #4 function key (highlighted)

TR1_OFF (--)

Turns off trigger 1.



Triggers topic

Arm Trig #1 function key (not highlighted)

TR2_OFF (--)

Turns off trigger 2.

 **Triggers** topic
Arm Trig #2 function key (not highlighted)

TR3_OFF (--)

Turns off trigger 3.

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

TR4_OFF (--)

Turns off trigger 4.

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

10.2 Trigger Conditions

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

TR1 (--)

Selects trigger 1.

TR2 (--)

Selects trigger 2.

TR3 (--)


Selects trigger 3.

TR4 (--)

Selects trigger 4.

=RX (--)

Triggers on events from the RX interface direction.

 **Triggers** topic
Trigger Conditions Menu
→ *Trigger Direction*
FROM RX function key

=TX (--)

Triggers on events from the TX interface direction.



Triggers topic

Trigger Conditions Menu

→ *Trigger Direction*

FROM TX function key

=BOTH (--)

Triggers on events from both interface directions (default).



Triggers topic

Trigger Conditions Menu

→ *Trigger Direction*

FROM BOTH function key

=PLAYBACK (--)

Triggers on events from capture RAM or disk playback.



Triggers topic

Trigger Conditions Menu

→ *Trigger Direction*

FROM PLAYBACK function key

Message and Parameter Triggers

The following commands are used to include/exclude messages and parameters in the currently selected trigger by identifying specific tokens. Refer to the appropriate Protocol Set Reference Manual for a list of usage symbols and valid tokens.

Tokens identified with a 'V' usage symbol and:

- a '?' (eg. variable length fields) can be used with the T_STRING command.
- and a '@' can be used with the T_VALUE command. Tokens which are used with fields formatted with point codes can also be used with the T_PC_VALUE command.

Tokens identified with a 'G' can be used with the T+ALL and T-ALL commands. Tokens identified with an 'F' can be used with the T+ and T- commands.

T+ (token --)

Sets the specified trigger.

T- (token --)

Clears the specified trigger.

T+ALL (token_group --)

Sets all valid triggers within the specified token group.

T-ALL (token_group --)

Clears all valid triggers within the specified token group.

T_VALUE (mask\value\token --)

Sets the trigger value for the specified token when the mask is 0; sets the trigger value to "don't care" when the mask is -1.

Example:

Set the CIC (circuit identification code) trigger value to "don't care".

```
-1 0 <CIC> T_VALUE
```

T_STRING (mask\string\token --)

Sets the trigger string for the specified token when the mask is 0; sets the trigger string to "don't care" when the mask is -1.

Example:

Set the digits trigger in the SCCP called party address to 4624545.

```
0 X" 4624545" <CDPA.ADDI> T_STRING.
```

T_PC_VALUE (mask sub fields\point code sub fields\token --)

Sets the trigger value for the point code formatted token. See Section 8 for point code formats.

Example:

Set the signalling point code trigger in the SCCP called party address to 3-45-1.

```
0 0 0 3 45 1 <CDPA.SPC> T_PC_VALUE
```

T+ALL_OTHERS (--)

Sets all tokens in undecoded protocol levels.

T-ALL_OTHERS (--)

Clears all tokens in undecoded protocol levels.

Complete levels of triggers can be reset to their initial state using the following commands. Tokens used to identify protocol levels are listed in Table 6-2.

T_RESET (token --)

Resets all trigger items in the specified protocol level.

T_RESET_ALL (--)

Resets all triggers.

TF_RESET (token --)

Resets all triggers and filters in the specified protocol level.

TF_RESET_ALL (--)

Resets all triggers and filters.



NOTE

TF_RESET and TF_RESET_ALL do not require a trigger or filter to be initially selected.

These reset commands do not affect routing labels. Refer to the RESET_ROUTING_LABELS command to reset labels to 0, and TURN_OFF_LABEL_TRIGGERS to turn off label triggers.

Routing labels (1 through 10) can be triggered using the following commands.

T+PC (label# --)

Sets the trigger for the specified routing label.

T-PC (label# --)

Clears the trigger for the specified routing label.

TURN_OFF_LABEL_TRIGGERS (--)

Clears all routing label triggers.

Example:

Clear trigger 1 routing labels.

TR1

TURN_OFF_LABEL_TRIGGERS

A specific string of characters on the received message can be used as a trigger condition.

=STRING (string --)

Specifies the string to trigger on. The string must be defined by enclosing the text characters in quotes or enclosing hex characters in quotes with a leading 'X' character. The maximum string length is 64 characters. The trigger condition is an anchored match from the first character in a received frame.

+STRING (--)

Sets the string trigger.

-STRING (--)

Clears the string trigger (default).

=MASK (string --)

Specifies the string mask. The string mask is used to specify "don't care" positions when matching strings. Each byte of the compare string and the received frame is ANDed with the corresponding byte of the string mask before they are compared.

Example:

Set the conditions for trigger 1 to execute when a frame with a length indicator of 60 is received.

TR1

X" 00003C" =MASK

X" 00003F" =STRING (check only length indicator bits)

+STRING

Other Triggers

+CAPT_FULL (--)

Sets the capture RAM trigger. The trigger is activated when the capture RAM is full.

-CAPT_FULL (--)

Clears the capture RAM trigger (default).

+DISK_FULL (--)

Sets the disk trigger. The trigger is activated when the disk recording file is full.

-DISK_FULL (--)

Clears the disk trigger (default).

=TIME (year\month\day\hour\minute --)

Specifies the time values for the time trigger. The trigger is activated when the specified minute is reached.

+TIME (--)

Sets the time trigger.

-TIME (--)

Clears the time trigger (default).

10.3 Trigger Actions

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

TA1 (--)

Performs the actions for trigger #1.

Example:

Set the action for trigger 1 to sound an audible alarm and start capture to RAM.

```
MAKE TA1 BEEP CAPT_ON ;
```

TA2 (--)

Performs the actions for trigger #2.

TA3 (--)

Performs the actions for trigger #3.

TA4 (--)

Performs the actions for trigger #4.

⚠ WARNING

If a trigger is armed, the action code will be reset as specified on the Trigger Action Menu. Ensure all trigger actions defined with the MAKE command are executed after the trigger is armed.

The frame or event which causes the trigger action to execute can be highlighted in the Data Display Window.

HIGHLIGHT (--)

Highlights the report in the Data Display Window.

HIGHLIGHT=RED (--)

Selects a red background for highlighting reports.

HIGHLIGHT=BLUE (--)

Selects a blue background for highlighting reports.

Example:

Set the action for trigger 2 to highlight the trigger event in blue.

```
MAKE TA2 HIGHLIGHT=BLUE HIGHLIGHT ;
```

10.4 Call Tracing

Triggers and filters in combination with the message decoder can be used for call tracing.

Example:

Use trigger and filter commands to set up and capture only ISUP messages related to a call to Called Party Number 4624545.

Trigger 2 action will start capture of the first IAM (initial address message) with the correct called party number (en-bloc sending) as well as all subsequent ISUP messages with the same CIC (circuit identification code). Trigger 1 action will stop capture when the next IAM is received with the same CIC. Since trigger 1 action is performed before trigger 2 action, if the next call on the same CIC happens to contain the correct called party number, trigger 1 actions will be overridden by trigger 2.

It is assumed that the trace monitor point is not between two STP's. If it is, then trigger 2 action can be enhanced to set trigger 1 and disk filter parameters on the correct OPC/DPC combinations.

```
TR1 T_RESET_ALL          ( switch control to trigger 1 and reset )
<CICX> T+                ( turn on CIC trigger )
```

```
TR2 T_RESET_ALL          ( switch control to trigger 2 and reset )
<IAM> T+ <CDPN> T+      ( turn on IAM and CDPN triggers )
<CDPN.NPLAN> T+ALL     ( set number plan to include all types )
<CDPN.INAI=SN> T+      ( set nature of address )
0 X" 4624545" <CDPN.ADDI> T_STRING ( set CDPN address digits )

OFF-LINE                ( turn off-line while changing trigger actions )
TR2_ON                  ( activate trigger 2 )

MAKE TA2                ( trigger 2 action )
<CIC> <>@                ( check CIC token value )
IF                       ( token exists? )
    TR1                  ( switch control to trigger 1 )
    D_FILTER             ( switch control to disk filter )
    0 SWAP 2DUP          ( set up CIC parameters )
    <CIC> T_VALUE        ( set CIC value trigger )
    <CIC> F_VALUE        ( set CIC value filter )
    <IAM> T+              ( turn on IAM trigger )
    <ISUP> F+            ( pass ISUP messages )
    " Capture started" W.NOTICE
ENDIF
;

TR1_ON                  ( activate trigger 1 )

MAKE TA1                ( trigger 1 action )
TR1                      ( switch control to trigger 1 )
D_FILTER                 ( switch control to disk filter )
<IAM> T-                 ( turn off IAM trigger )
<ISUP> F-                 ( block ISUP messages to disk )
" Capture stopped" W.NOTICE
;

D_FILTER                ( switch control to disk filter )
F_RESET_ALL             ( reset filters )
<FISU> F-                ( block FISU's )
<LSSU> F-                ( block 1 byte LSSU's )
<LSS2> F-                ( block 2 byte LSSU's )
<SNTM> F-                ( block SNTM's )
<SNMM> F-                ( block SNMM's )
F-ALL_OTHERS            ( block all undecoded messages )
<CIC> F+                 ( pass CIC parameters )
<ISUP> F-                 ( block ISUP messages )
ACTIVATE_DISK           ( activate disk filter )
RECORD WD3:CALL_TRACE   ( start disk recording )
ON-LINE
```

11

LEVEL 1 DECODER

The level 1 decoder decodes physical events and returns a value corresponding to the event type on the Primary Rate interface.

L1-ID@ (-- value)

Contains an identifier for the level 1 event. Possible values are:

R#NORMAL	Synchronized signal
R#REDALM	Red alarm
R#YELALM	Yellow alarm
R#LSTSIGL	Lost signal
R#LSTPLOCK	Lost phase lock
R#OOFALM	Out of frame alarm

In addition, the CEPT-30 framing format can produce the following values:

R#RMYELA	Port A Multiframe yellow alarm
R#RMYELB	Port B Multiframe yellow alarm
R#PCMMNMLA	Port A Normal multiframe condition
R#PCMMNMLB	Port B Normal multiframe condition

END-TIME (-- address)

Contains the 48 bit timestamp for the event. END-TIME can be used with the GET_TSTAMP_MILLI or GET_TSTAMP_MICRO commands as described in the Programmer's Reference Manual.

BLOCK-COUNT (-- address)

Contains the block sequence number for the level 1 event. Every received event is assigned a unique sequence number.

12

SIMULATION ARCHITECTURE

The SS#7 Simulation program is a combination of the complete SS#7 Monitor application, together with a partial emulation for level 2.

SS#7 Simulation provides basic mechanisms for simulation at levels 2, 3, and 4. These mechanisms include FISU idling, auto sequence numbering, level 2 timer support, and functions for setting up and sending messages.

12.1 Live Data

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

The test manager, if active, processes the received data and generates a response. Finally, outgoing data is framed by the level 2 simulation and sent out to the physical interface.

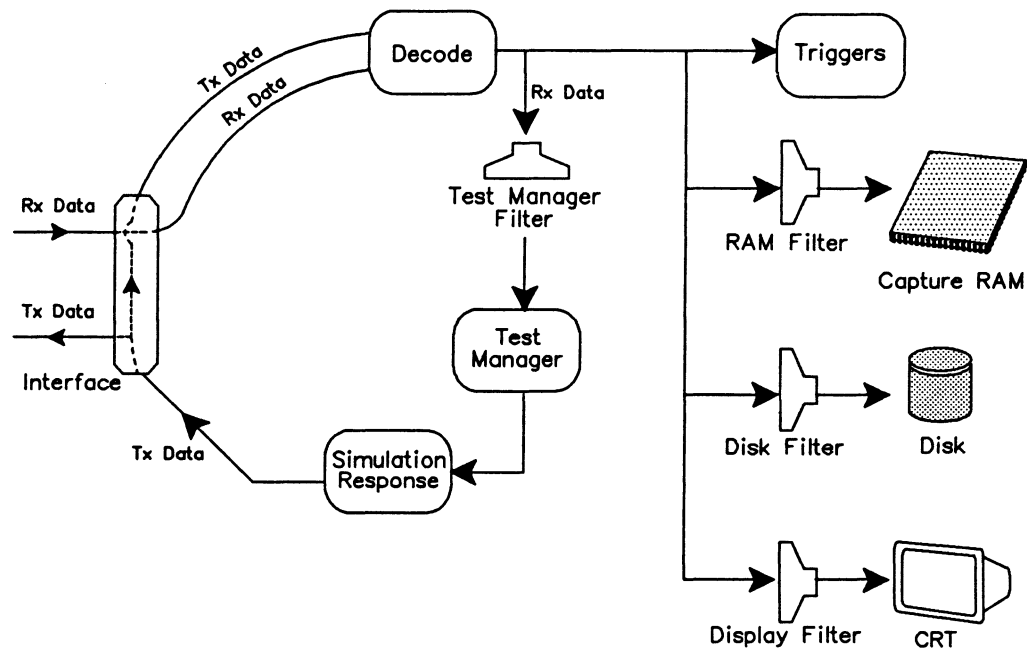


Figure 12-1 SS#7 Simulation Data Flow Diagram – Live Data

 **Display topic**
Live Data function key

MONITOR (--)

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

12.2 Playback

Simulation through the test manager can be run concurrently with playback from disk or capture RAM. The simulation program can multiplex data coming in from different sources. Figures 12-2 and 12-3 show the separate paths which playback and simulation take in the application.

Data, played back from disk or capture RAM, proceeds through the decode sections, triggers, filters, and finally the display. At the same time, live data is processed as described in Section 12.1.

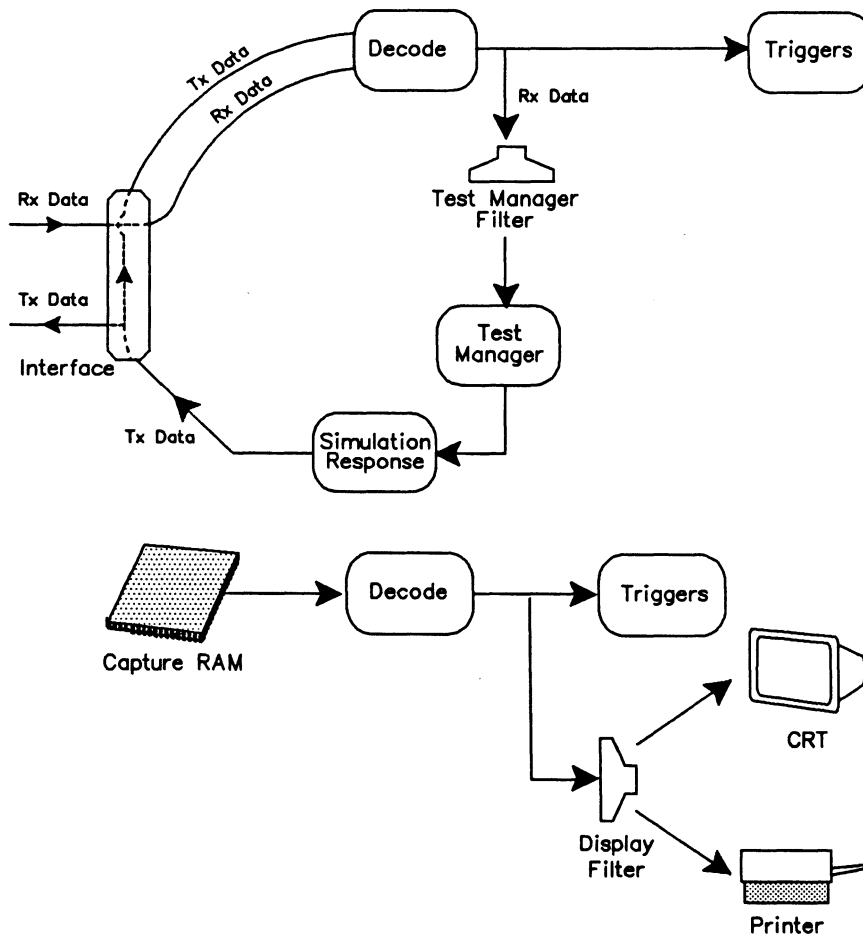


Figure 12-2 SS#7 Simulation Data Flow Diagram - Offline Processing

 FROM_CAPT HALT
Display topic
 Playback RAM function key

 FROM_DISK HALT PLAYBACK
Display topic
 Playback Disk function key

HALT (--)

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

12.3 Simultaneous Live Data and Playback

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

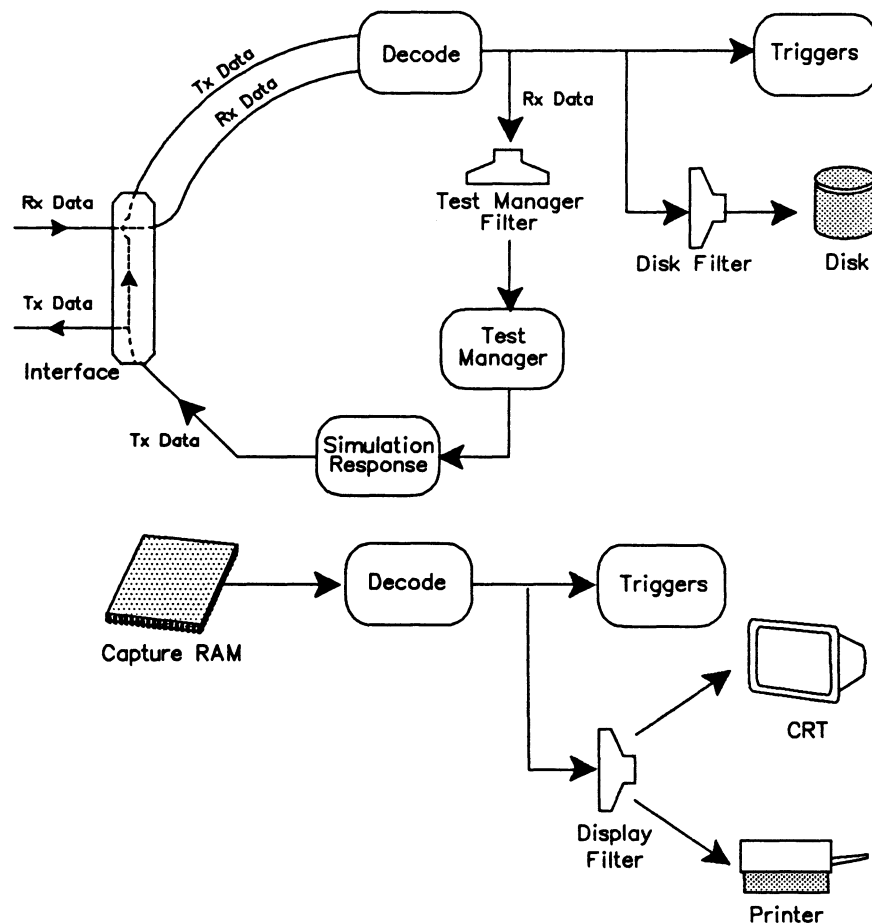


Figure 12-3 SS#7 Simulation Data Flow Diagram – Freeze Mode

 RECORD FREEZE FROM_CAPT

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.

13

SIMULATION CONFIGURATION

This section describes the commands associated with each item on the Level 1 and Level 2 Configuration Menus and the Protocol Set Selection Menu.

13.1 Level 1

This section describes the commands to configure the physical interface and route data to the appropriate application processor or output device for WAN (wide area network) or PRA (primary rate access).

The Primary Rate interface is configured on the Home processor prior to loading the application.

The WAN interface is configured on the application processor after loading and switching to the SS#7 Simulation.

Level 1 Configuration Menu	
Signalling Data Link Level	
→ Interface Mode	DTE
Interface Type	RS232C/V.28
Clocking	NRZ WITH CLOCK
Bit Rate	64000
BOF Timestamp	OFF

Figure 13-1 Level 1 Simulation Configuration Menu

Signalling Data Link Level

→ *Interface Mode*

=SIM_DCE (--)

Selects the DCE interface mode.

 DCE function key

=SIM_DTE (--)


Selects the DTE interface mode (default).

 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 connector and electrically isolates the other connectors on the port.

 *RS422C/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 connector and electrically isolates the other connectors on the port.

 *RS449/V.36* function key

→ *Clocking*

CLK=STD (--)

Selects the standard non-return to zero line encoding (default).

 *NRZ WITH CLOCK* function key

CLK=EXT_CLK (--)

Provides a transmit by the DTE clock on pin 24 of an RS-232C connector.

 *EXTERNAL TX CLOCK* function key

→ *Bit Rate (WAN and PRA Interface)*

The interface speed can be selected from preset values, set to a user-defined speed, or measured depending on the emulation interface and clocking selections.

Interface	Clocking	Action
DCE WAN	NRZ WITH CLOCK	Select
DTE WAN	NRZ WITH CLOCK	Measure
DCE WAN	EXTERNAL WITH CLOCK	Measure
DTE WAN	EXTERNAL WITH CLOCK	Select
PRA	---	Measure

SPEED! (speed --)

Sets the interface speed to the specified value.

 *Modify Speed* function key

SPEED@ (-- speed)

Returns the current interface speed.

 *Measure Speed* function key

→ *BOF Timestamp* (WAN and PRA Interface)

BOF-TSTAMP! (flag --)

Specifies whether beginning of frame timestamps are recorded on incoming messages (default is off).

 *ON/OFF* function key

The following commands control the flow of data to the application program.

OFF-LINE (--)

Turns the receiver off (default). The transmitter continuously sends marks.

 **Simulation** topic
Online function key (not highlighted)

ON-LINE (--)

Turns the receiver on. The transmitter continuously sends flags if FISU idling is off, and FISU's if FISU idling is on (see Section 13.2).

 **Simulation** topic
Online function key (highlighted)

The following commands control WAN interface leads.

LEAD_OFF (lead identifier --)**LEAD_ON (lead identifier --)**

Turns the specified lead on or off. The complete list of lead identifiers is contained in the Programmer's Reference Manual.

LEAD_CHECK (lead identifier -- flag)

Returns the current state of the specified lead (on or off). If the lead is invalid for the current interface mode and type, -1 is returned.

13.2 Level 2

This section describes the commands and variables to configure level 2 of the SS#7 Simulation.

Level 2 Configuration Menu			
→ SU Compression	MAX	Signalling Link Level FISU Idling	OFF
SUERM Function	ON	Octet Counting Mode	AUTO
T Threshold	64	Maximum SIF	272
Initial Alignment Counters			
Tin Threshold 4	Tie Threshold 1	M	Max Retry 5
Timers (10th secs)			
T1 410	T3 12	T4e 5	T6 46
T2 750	T4n 82	T5 1	T7 13

Figure 13-2 Level 2 Simulation Configuration Menu

Signalling Link Level

→ *SU Compression*

COMPRESS-SU! (value --)

Sets the number of identical FISU's or LSSU's to compress on live data to a specified value. The SU_MAX value sets the maximum compression at 99,999,999 (default). The OFF value sets the compression to off.

 *Modify Count/MAX/OFF* function key


COMPRESS-SU@ (-- value)

Returns the current SU compression ratio.

→ *SUERM Function*

SUERM-FUNCTION! (flag --)

Selects whether the SUERM (signal unit error rate monitor) is active (default is off). When turned on, the error counter is reset to 0.

 *ON/OFF* function key

SUERM-FUNCTION@ (-- value)

Returns true if the SUERM is active.

→ *T Threshold*

SUERM-THRESH! (value --)

Sets the SUERM threshold for determining a link failure indication. Valid values are 1 through 65535 (default is 64).

 *Modify Value* function key

SUERM-THRESH@ (-- value)

Returns the current value of the SUERM threshold.

SUERM-COUNT! (value --)

Sets the SUERM error counter. Valid values are 0 through 65535.

SUERM-COUNT@ (-- value)

Returns the current value of the SUERM error counter.

→ *FISU Idling*

AUTO_FISU (flag --)

Selects whether FISU's are automatically sent whenever the transmit channel is idle (default is off). The forward and backward sequence bytes are set to equal the values in the most recently sent buffer. By default, transmitted FISU's have BSN and FSN equal to 127, and BIB and FIB equal to 1.

 *ON/OFF* function key

**NOTE**

With FISU idling off, flags are automatically sent whenever the transmit channel is idle.

→ *Octet Counting Mode*

OCTET-MODE! (mode --)

Selects the octet counting mode used by the simulation to ON, OFF (default), or AUTO.

 *ON/OFF/AUTO* function key

OCTET-MODE@ (-- mode)

Returns the current state of the octet counting function.

→ *Maximum SIF*

SIF-MAXSIZE! (value --)

Specifies the maximum SIF (signalling information field) size to turn octet counting off when the octet counting mode is set to AUTO. Valid values are 1 through 999 (default is 272).

 *Modify SIF* function key

SIF-MAXSIZE@ (-- value)

Returns the current value of the maximum SIF size.

Initial Alignment Counters

AERM-FUNCTION! (flag --)

Selects whether the AERM (initial alignment error rate monitor) is on (default is off). If on, *SUERM function* is automatically disabled.

AERM-FUNCTION@ (-- flag)

Returns true if the AERM function is on.

→ *Tin Threshold*

AERM-N-THRESH! (value --)

Specifies the normal AERM threshold for determining unsuccessful initial alignment. Valid values are 1 through 999 (default is 4).

 *Modify Value* function key

AERM-N-THRESH@ (-- value)

Returns the normal AERM threshold value.

→ *Tie Threshold*

AERM-E-THRESH! (value --)

Specifies the emergency AERM threshold for determining unsuccessful initial alignment. Valid values are 1 through 999 (default is 1).

 *Modify Value* function key

AERM-E-THRESH@ (-- value)

Returns the emergency AERM threshold value.

→ *Max Retry*

MAX-RETRY! (value --)

Specifies the maximum number of retries for determining unsuccessful initial alignment. Valid values are 1 through 999 (default is 5).

 *Modify Value* function key

MAX-RETRY@ (-- value)

Returns the maximum number of retries for determining unsuccessful initial alignment.

Timers (10th secs)**START_SS7_TIMER (timer# --)**

Starts the specified timer. Valid level 2 timers are listed in the following table.

Timer #	Description
L2#T1	Timer T1 (Timer "aligned/ready")
L2#T2	Timer T2 (Timer "not aligned")
L2#T3	Timer T3 (Timer "aligned")
L2#T4	Timer T4 (Proving period timer)
L2#T4N	Timer T4n (Normal proving period timer)
L2#T4E	Timer T4e (Emergency proving period timer)
L2#T5	Timer T5 (Timer "sending SIB")
L2#T6	Timer T6 (Timer "remote congestion")
L2#T7	Timer T7 (Timer "excessive delay of acknowledgement")

Table 13-1 Simulation Timers

SS7_TIMER@ (timer# -- value)

Returns the value, in tenths of seconds, of the specified simulation timer.

SS7_TIMER! (timer#/value --)

Sets the simulation timer to the specified value.

 *Modify Value* function key

SS7_TIMER_DEFAULT@ (timer# -- value)

Returns the default value of the specified simulation timer. In some cases, the default value depends on the bit rate of the signalling link.

Example:

Set timer T6 to the default.

```
L2#T6 DUP SS7_TIMER_DEFAULT@ SS7_TIMER!
```

 *Default* function key

PE->T4 (--)

Sets timer T4 equal to timer T4e.

PN->T4 (--)

Sets timer T4 equal to timer T4n.

?T4=PE (-- flag)

Returns true if timer T4 equals timer T4e.

13.3 Configuration Storage

The current settings of the level 1 and level 2 simulation configurations can be saved to disk for future retrieval.

SAVE_CONFIG (filename --)

Saves the current configuration.

Example:

Save the current configurations in a file named 'Config' on drive DR0.

```
" DR0:Config" SAVE_CONFIG
```



Simulation topic

Save Config function key

LOAD_CONFIG (filename --)

Loads a previously saved configuration.



Simulation topic

Load Config function key

13.4 Protocol Set Selection

The decoder can be configured to use a particular set of protocol files. Refer to the appropriate Protocol Set Reference Manual for valid filenames.

LOAD_PROTOCOL_SET (filenames\number --)

Loads the specified protocol file(s). In addition, playback from a disk file is stopped and the display format, trigger, filter, and routing label settings are restored to the default configurations.

Example:

Load the CCITT TUP protocol with supporting lower layers.

```
" CCITT_TUP88.T" "CCITT_NET88.T" "CCITT_LINK88.T" 3 LOAD_PROTOCOL_SET
```



Simulation topic

Protocol Set Selection Menu

→ *CCITT_TUP88*

Select function key

→ *CCITT_NET88*

Select function key

→ *CCITT_LINK88*

Select function key

Load Protocols function key



NOTE

LOAD_PROTOCOL_SET should be used before loading test scripts. When a protocol set is loaded, test scripts are cleared.

 **WARNING**

Colon definitions that compile message tokens will crash if the protocol set is changed after creating the colon definition.


SELECT_VAR (protocol set --)

Selects the specified protocol set.

 Protocol Variance Menu
Select function key

LOAD_ALL (--)

Loads all functional parts of the protocol set selected with SELECT_VAR.

 Protocol Set Selection Menu
All function key
Load function key

Example:

Load all available CCITT 1988 functional parts.

```
" CCITT_88" SELECT_VAR      ( Selects the CCITT 1988 protocol set )  
LOAD_ALL                    ( Loads all functional parts )
```


14

LEVEL 2 SIMULATION

This section describes the commands used during level 2 simulation.

14.1 Sequence Numbering

FSNT! (number --)

BSNT! (number --)

FIBT! (number --)

BIBT! (number --)

Specifies a number to be used on the next signal unit transmitted via the SEND_BUFFER_SU or SEND_LSSU command. The numbers correspond to the forward sequence number, backward sequence number, forward indicator bit, and backward indicator bit. The default value is 127 for FSNT! and BSNT!, and 1 for FIBT! and BIBT!.

FSNT@ (-- number)

BSNT@ (-- number)

FIBT@ (-- number)

BIBT@ (-- number)

Returns a number to be used on the next signal unit transmitted via the SEND_BUFFER_SU command.



NOTE

If AUTO_FSN is set, the FSN used in the FSNT! and FSNT@ commands will be 1 less than in the value contained in the next transmitted MSU.

AUTO_FSN (flag --)

Selects whether the FSN (forward sequence number) is automatically incremented (modulo 128) before an MSU is transmitted via the SEND_BUFFER_SU command.

14.2 Send Commands

Up to 256 buffers are available to create and send messages. These buffers are numbered from 0 through 255. All buffers are cleared by the TCLR command.

The buffer structure consists of 4 reserved bytes, followed by 2 bytes containing the 16 bit length, followed by the signal unit. The value for the forward and backward sequence bytes is not set until the buffer is transmitted. At that time, the buffer's forward and backward sequence bytes are set according to the values set via the FSNT, BSNT, FIBT, and BIBT commands (see Section 14.1).

The buffer does not contain the CRC bytes; instead, these bytes are attached when the buffer is transmitted.

There are three methods of moving text into a buffer.

Methods 1 and 2 automatically allocate memory for the specified text (stored from the third octet of the signal unit). Method 3 requires the user to allocate memory before moving text into the buffer (stored from the first octet of the signal unit). Use the TCLR command to clear all buffers.

AUTO_LI (flag --)

Selects whether the length indicator byte is set automatically according to the length of the signal unit.



NOTE

Use with Methods 1 and 2 only.

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 )
```

APPEND->BUFFER (string --)

Appends a quote string to the most recently created buffer. This allows messages to be created with a total length greater than 80 characters. The length indicator is automatically adjusted if AUTO_LI is set.

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
```

HFILE->BUFFER (filename\buffer number --)

Similar to the FILE->BUFFER command except only ASCII characters '0' through '9', 'a' through 'f', and 'A' through 'F' are allowed. The characters are translated to hexadecimal. This file can be terminated with the ASCII character 'x' or CTRL 'p'.

Method 3

The following four commands should not be used with Methods 1 and 2.

ALLOT_BUFFER (size \ buffer number -- flag)

Allocates memory for the specified buffer. ALLOT_BUFFER returns 0 if an error occurred, or 1 if correct.



NOTE

ALLOT_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 $!           ( Initialize the string )
16 3 ALLOT_BUFFER                   ( Allocate 16 bytes of memory )
IF
    tempstring 4+ 5 3 FILL_BUFFER    ( Move 'TEST ' to buffer )
    " FAIL" COUNT 3 APPEND_TO_BUFFER ( Append 'FAIL' 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

SEND_BUFFER_SU (attribute\buffer number --)

Sends the indicated buffer using the specified attribute. The forward and backward sequence bytes are modified according to values set with FSNT, BSNT, FIBT, and BIBT. Zero bit insertion is performed on the transmitted buffer. The attribute can be one of the following:

CRC	Sends buffer with the correct CRC attached.
CRC_ERR	Sends buffer with an invalid CRC attached.
ABT	Aborts buffer after sending 5 bytes of the buffer (eg. sends 7 continuous one's without zero bit insertion).
REPT	Sends buffer repeatedly until another buffer is sent.

The attributes can be OR'd together before executing the SEND_BUFFER_SU command. If the attribute does not contain REPT and FISU idling is off, flags are continuously transmitted after sending the message. If the attribute does not contain CRC_ERR, the CRC attribute is implied.

Example:

Send buffer 1 repeatedly with the correct CRC value.

```
REPT 1 SEND_BUFFER_SU
```

SEND_LSSU (attr\type --)

Sends an LSSU using the indicated attribute and type. The type can be any of the following constants:

<SIO>#	Status indication "O"
<SIN>#	Status indication "N"
<SIE>#	Status indication "E"
<SIOS>#	Status indication "OS"
<SIPO>#	Status indication "PO"
<SIB>#	Status indication "B"

The attributes are the same as described for the SEND_BUFFER_SU command.

SF_SIZE (1|2 --)

Selects the size, in bytes, of LSSU's sent with the SEND_LSSU command (default is 1).

CURRENT_LSSU (-- type)

Returns the type of LSSU most recently sent via the SEND_LSSU command.

SEND_MARKS (--)

Sends the mark character continuously.

SEND_SPACES (--)

Sends the space character continuously.

SEND_FLAGS (--)

Sends flags continuously.

SEND_FISU (attribute --)

Sends an FISU using the specified attribute. Refer to the SEND_BUFFER_SU command for valid attributes. The attribute can be one of the following:

CRC	Sends FISU with the correct CRC attached.
CRC_ERR	Sends FISU with an invalid CRC attached.
REPT	Sends FISU repeatedly until another buffer is sent.

15

TEST MANAGER

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 SS#7 Monitor/Simulation application.

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

This section reviews basic ITL components and describes the protocol event and action commands specific to SS#7.

15.1 ITL Constructs

Following is a brief description of test manager constructs. For more details and examples, refer to the Programmer's Reference Manual.

TCLR (--)

Initializes the test manager. Any existing test suites already in memory are cleared. The current state is set to 0. All test scenarios should start with the TCLR command.

STATE_INIT{ }STATE_INIT (number --)

Brackets the execution sequence performed prior to entering a state. The initialization logic for a state is executed independently of how it was called.

This initialization procedure can be used for any state but is not compulsory. STATE_INIT{ } must be preceded by the number of the state being initialized, eg. 0 STATE_INIT{ }.

If the ACTION{ }ACTION sequence does not result in a change of state, the STATE_INIT{ }STATE_INIT is not re-executed.

STATE{ }STATE (number --)

Brackets a state definition. STATE{ } must be preceded by the number of the state. Valid values are 0 through 255. State 0 must be defined within an ITL program. If not, the test manager will not run the script. If multiple states are defined with the same number in the test script, the test manager uses the latest definition.

ACTION{ }ACTION (flag --)

Brackets the set of tasks, decisions, and outputs which execute once the expected event is received by the test manager. There must be at least one action defined for each expected event. The action is executed when the flag is true (non-zero).

NEW_STATE (number --)

Executes the initialization logic of the specified state (providing STATE_INIT{ }STATE_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{ }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 from a state subroutine call.

NEW_TM (filename --)

Loads and compiles the specified file and then starts the test manager at state 0. It can be included as part of the action field to load and execute another scenario.

CLEAR_KEYS (--)

Clears all the test keys.

COUNTER1 ... 128 (-- address)

There are 128 general purpose variables, named COUNTER1 ... COUNTER128, which can be used by the test manager.

15.2 Event Recognition

During test script execution, any event received by the test manager is evaluated to determine if it matches the event-specifier of the first action within that state. If the evaluation does not return a true value, the following action clauses are evaluated in a sequential manner. Once an event evaluates true, the subsequent action clauses in that particular state are not examined.

Level 1

?L1_EVENT (value -- flag)

Returns true if the specified level 1 event has occurred. The Primary Rate event identifiers are:

R#NORMAL	Synchronized signal
R#REDALM	Red alarm
R#YELALM	Yellow alarm
R#LSTSIGL	Lost signal
R#LSTPLOCK	Lost phase
R#OOFALM	Out of frame

In addition, the CEPT-30 framing format event identifiers are:

R#RMVELA	Port A Multiframe yellow alarm
R#RMVELB	Port B Multiframe yellow alarm
R#PCMMNMLA	Port A Normal multiframe condition
R#PCMMNMLB	Port B Normal multiframe condition

Events can be OR'd together before executing the ?L1_EVENT command.

Example:

```
R#REDALM R#YELALM OR ?L1_EVENT
```

```
ACTION{
```

```
    .
    code to be executed when the Primary Rate interface is in red or yellow alarm
    .
```

```
}ACTION
```

L1_EVENT (-- flag)

Returns true if any level 1 event has occurred.

Level 2

?L2_EVENT (value -- flag)

Returns true if the specified level 2 simulation event has occurred. The level 2 event identifiers are:

LINK_FAILURE#
LINK_IN_SERVICE#
OCTETMODE_ON#
OCTETMODE_OFF#

L2_EVENT (-- flag)

Returns true if any level 2 simulation event has occurred.

?EOF_IND (-- flag)

Returns true if the event is a received frame.

The following three commands can be used to test the presence and error status of fields in a received message. These fields are identified with the tokens listed in the appropriate Protocol Set Reference Manual. All tokens can be used with the commands. Additionally, a wildcard token '<*>' can be used to represent any token.

?RX_FRAME (token -- flag)

Returns true if the event is a received frame and the decoded frame contains the specified token.

?RX_ERROR (token\error -- flag)

Returns true if the event is a received frame and the decoded frame contains an error of the type specified. Errors are specified using a combination of tokens and the following error identifiers.

General errors:

ANY#	Any error (including token specific errors)
CRC#	CRC error
ABORT#	Abort error (i.e. seven consecutive one's)
RX-OVFL#	Receiver overflow
FRAMING#	Framing error (eg. flag not octet aligned)
RX-LONG#	Frame truncated to 1004 bytes

Token specific errors:

TOKEN#	Any token specific error
SHORT#	Frame terminated before decode process completed. Error associated with last decoded token.
LONG#	Frame contains unexpected data following successful completion of the decode process. Error associated with last decoded token.
VIOLATION#	Token does not conform to the specification
EXTENSION#	Spare bit token not coded with zeros

Error identifiers can be OR'd together. They can also be inverted using the NOT command to indicate errors other than the ones indicated.

A token is specified to restrict the search of errors to the identified token.

Example:

```
<*> CRC# ?RX_ERROR          ( Returns true if CRC error received )
<LUN> CRC# ?RX_ERROR        ( Returns true if link uninhibit message is received )
                               ( with a CRC error )
<*> VIOLATION# EXTENSION# OR ?RX_ERROR  ( Returns true if any message is received )
                                       ( with either a violation or extension )
                                       ( error )
<MIM> ANY# EXTENSION# NOT AND ?RX_ERROR ( Returns true if a management inhibit
                                       ( message is received with any error
                                       ( except extensions )
```

?RX_GOOD (token -- flag)

Returns true if the event is a received frame, the decoded frame contains the specified token, and there are no general errors or errors on the token.

?RX_GOOD is equivalent to the following sequence:

```
DUP ?RX_FRAME SWAP ANY# ?RX_ERROR 0= AND
```

?RECEIVED (string -- flag)

Returns true if a user-defined character string is found in the received frame.

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



NOTE

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

?SEARCH (string -- flag)

Returns true if a user-defined character string is found in the received frame.

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

Example:

Search for the string 'IDACOM' which could be located starting at any position within the received frame.

```
" IDACOM" ?SEARCH
```

?ABORT (-- flag)

Returns true if an abort frame is received.

?CRC_ERROR (-- flag)

Returns true if a frame with a CRC error is received.

Other Events

?TIMER (timer_number -- flag)

Returns true if the specified timer has expired. Out of a total of 128 system timers, timers 1 to 100 are available in monitor mode, and timers 1 to 64 are available in simulation mode.

TIMEOUT (-- flag)

Returns true if any timer has expired.

?MAIL (-- flag)

Returns true if mail has been received from one of the other applications.

?KEY (key -- flag)

Returns true if the specified function key has been pressed.

OTHER_EVENT (-- true flag)

Returns true so that an ACTION{ }ACTION statement can check for events that have not been explicitly programmed into an event field.

EVENT-TYPE@ (-- value)

Returns an event identifier. Valid event identifiers are:

LEVEL1#	Level 1 event identified by L1-ID@ has been received
LEVEL2#	Level 2 simulation event has been received
FRAME#	Level 2 frame has been received
REPEAT#	The application has started to compress frames
TIME-OUT#	Timer identified by the TIMER-NUMBER variable has expired
FUNCTION-KEY#	Function key identified by the KEY-NUMBER variable has been pressed
COMMAND#	Interprocessor mail has been received and can be retrieved through the EXTRACT_FTH_DATA command

Example:

```
OTHER_EVENT
ACTION{
    EVENT-TYPE@ TIME-OUT# =
    IF
        .
        code to be executed if a timer has expired
        .
    ENDIF
}ACTION
```

16

TEST SCRIPTS

This section contains sample test scripts. These tests have also been supplied on disk and can be loaded as described in the Programmer's Reference Manual.

16.1 LINK_UP.F

The LINK_UP.F test script simulates one side of the initial alignment process used to bring a link from the out-of-service state. The simulation is started via function keys or upon receipt of an SIE or SIO LSSU.

```
( ----- )
( File Title:  LINK_UP.F )
( Version:    1.5 )
(           This script simulates an SS#7 switch in order to bring )
(           up a link. )
( ----- )

TCLR                ( initialize the test manager )

#IFNOTDEF LINK_UP.F ( if this script has not yet been loaded, )
                   ( the variables have to be compiled. )

0 CONSTANT LINK_UP.F ( signature word for this file )

NO VARIABLE EMERGENCY ( local processor in emergency state? )
NO VARIABLE FURTHER-PROVING ( further proving required? )

#ENDIF

1 SEQ{ ( -- ) ( start further proving )
  YES AERM-FUNCTION! ( restart the AERM )
  NO FURTHER-PROVING ! ( cancel further proving )
  L2#T4 START_SS7_TIMER ( restart "proving" timer )
}SEQ

2 SEQ{ ( -- ) ( set emergency proving )
  L2#T4 STOP_TIMER ( stop "proving" timer )
  PE->T4 ( set proving period to emergency )
  NO AERM-FUNCTION!
  1 RUN_SEQ ( start further proving )
}SEQ
```

```

3 SEQ{ ( -- ) ( stop alignment )
  NO EMERGENCY ! ( cancel emergency, if marked )
  CRC CURRENT_LSSU SEND_LSSU
  0 NEW_STATE
}SEQ

4 SEQ{ ( set state 2 conditions )
  COMPRESS-SU@ ( save current SU compression on stack )
  NO COMPRESS-SU! ( turn SU compression off temporarily )
  COMPRESS-SU! ( reset SU compression to previous value )
  EMERGENCY @ ( in emergency state? )
  IF
    PE->T4 ( set proving period to emergency )
    REPT <SIE> <># SEND_LSSU ( send an emergency alignment ind. )
  ELSE ( not in emergency state? )
    <SIE> ?RX_FRAME ( SIE received? )
    IF
      PE->T4 ( set proving period to emergency )
    ELSE ( everything normal? )
      PN->T4 ( set proving period to normal )
    ENDIF
    REPT <SIN> <># SEND_LSSU ( send a normal alignment ind. )
  ENDIF
  L2#T3 START_SS7_TIMER ( start timer T3 )
}SEQ

```

(***** STATE MACHINE *****)

```

0 STATE_INIT{
  NO AUTO_FISU ( stop all traffic )
  NO SUERM-FUNCTION! ( cancel SUERM function )
  NO EMERGENCY ! ( cancel the emergency state )
  CLEAR_KEYS ( clear key labels )
  " Align" 1 LABEL_KEY ( key to initiate initial alignment )
  " Emergency" 2 LABEL_KEY ( key to use emergency proving periods )
}STATE_INIT

```

```

0 STATE{ ( idle state )
  <SIO> ?RX_FRAME <SIE> ?RX_FRAME OR <SIN> ?RX_FRAME OR
  ACTION{ ( automatic start mode )
    4 RUN_SEQ
    1 CLEAR_KEY
    " Stop" 1 LABEL_KEY ( key to stop initial alignment )
    2 NEW_STATE
  }ACTION

```



```

UF1 ?KEY                ( start initial alignment procedure? )
ACTION{
  REPT <SIO> <># SEND_LSSU    ( send out an alignment indication )
  L2#T2 START_SS7_TIMER      ( start timer T2 )
  1 CLEAR_KEY
  " Stop"          1 LABEL_KEY ( key to stop initial alignment )
  1 NEW_STATE       ( GOTO Not Aligned state )
}ACTION

UF2 ?KEY                ( set proving periods to emergency? )
ACTION{
  YES EMERGENCY !          ( mark that currently in emergency )
  2 CLEAR_KEY             ( clear the emergency key )
}ACTION
}STATE

1 STATE{                ( Not aligned state )
  <SIO> ?RX_FRAME <SIE> ?RX_FRAME OR <SIN> ?RX_FRAME OR
  ACTION{
    L2#T2 STOP_TIMER        ( stop timer T2 )
    4 RUN_SEQ
    2 NEW_STATE             ( GOTO Aligned state )
  }ACTION

  UF2 ?KEY                ( set proving periods to emergency? )
  ACTION{
    YES EMERGENCY !          ( mark that currently in emergency )
    2 CLEAR_KEY             ( clear the emergency key )
  }ACTION

  UF1 ?KEY                ( stop initial alignment? )
  ACTION{
    L2#T2 STOP_TIMER        ( stop timer T2 )
    3 RUN_SEQ              ( stop alignment procedure )
  }ACTION

  L2#T2 ?TIMER            ( timer T2 expired? )
  ACTION{
    T." Alignment not possible." TCR ( report alignment error. )
    3 RUN_SEQ              ( stop alignment procedure )
  }ACTION
}STATE

2 STATE{                ( aligned state )
  <SIE> ?RX_FRAME
  ACTION{
    PE->T4                  ( set timer T4 to T4E )
    3 NEW_STATE
  }ACTION
}STATE

```

```
<SIN> ?RX_FRAME
ACTION{
  3 NEW_STATE
}ACTION

<SIOS> ?RX_FRAME L2#T3 ?TIMER OR
ACTION{
  T." Alignment not possible." TCR ( report alignment error. )
  3 RUN_SEQ ( stop alignment procedure )
}ACTION

UF2 ?KEY ( set proving periods to emergency? )
ACTION{
  REPT <SIE> <># SEND_LSSU
  PE->T4 ( set timer T4 to T4E )
  2 CLEAR_KEY ( clear emergency key )
}ACTION

UF1 ?KEY ( stop initial alignment? )
ACTION{
  L2#T3 STOP_TIMER ( stop timer T3 )
  3 RUN_SEQ ( stop alignment procedure )
}ACTION
}STATE

3 STATE_INIT{
  L2#T3 STOP_TIMER ( stop timer T3 )
  YES AERM-FUNCTION! ( start AERM simulator )
  L2#T4 START_SS7_TIMER ( start "proving" timer )
  0 COUNTER1 ! ( reset abort counter )
  NO FURTHER-PROVING ! ( cancel further proving )
}STATE_INIT

3 STATE{
  ( Proving state )
  L2#T4 ?TIMER ( proving period has expired? )
  ACTION{
    FURTHER-PROVING @ ( further proving required? )
    IF
      1 RUN_SEQ ( start further proving )
    ELSE
      NO AERM-FUNCTION! ( turn off the AERM )
      T." Alignment complete." TCR ( notify user of alignment complete )
      4 NEW_STATE ( GOTO Aligned Ready state )
    ENDIF
  }ACTION

<SIOS> ?RX_FRAME
ACTION{
  L2#T4 STOP_TIMER ( stop "proving" timer )
  NO AERM-FUNCTION! ( turn off the AERM )
  T." Alignment not possible." TCR ( report alignment error. )
  3 RUN_SEQ ( stop alignment procedure )
}ACTION
```

```

<SIO> ?RX_FRAME
ACTION{
  L2#T4 STOP_TIMER           ( stop "proving" timer )
  NO AERM-FUNCTION!         ( turn off the AERM )
  L2#T3 START_SS7_TIMER     ( start "aligned" timer )
  2 NEW_STATE               ( GOTO Aligned state )
}ACTION

<SIE> ?RX_FRAME
ACTION{
  ?T4=PE 0=                 ( proving period set to emergency? )
  IF
    2 RUN_SEQ               ( move to emergency proving state )
  ENDIF
}ACTION

UF1 ?KEY                    ( stop initial alignment? )
ACTION{
  L2#T4 STOP_TIMER         ( stop "proving timer" )
  NO AERM-FUNCTION!       ( turn off the AERM )
  3 RUN_SEQ               ( stop alignment procedure )
}ACTION

LINK_FAILURE# ?L2_EVENT    ( link failure indicated? )
ACTION{
  1 COUNTER1 +!           ( implies AERM Threshold exceeded )
  COUNTER1 @ MAX-RETRY@ = ( increment the number of aborts )
  IF
    L2#T4 STOP_TIMER     ( stop "proving" timer )
    NO AERM-FUNCTION!   ( turn off the AERM )
    T." Alignment not possible." TCR ( report alignment error. )
    3 RUN_SEQ           ( stop alignment procedure )
  ELSE
    YES FURTHER-PROVING ! ( further proving is required )
  ENDIF
}ACTION

UF2 ?KEY
ACTION{
  REPT <SIE> <># SEND_LSSU ( send repeating LSSUs )
  2 RUN_SEQ               ( move to emergency proving state )
  2 CLEAR_KEY            ( clear the emergency key )
}ACTION

<LI> ?RX_GOOD              ( correct SU received? )
ACTION{
  FURTHER-PROVING @      ( further proving required? )
  IF
    1 RUN_SEQ            ( start further proving )
  ENDIF
}ACTION
}STATE

```

```

4 STATE_INIT{
  YES AUTO_FISU                ( send FISUs )
  YES SUERM-FUNCTION!         ( start the SUERM )
  CLEAR_KEYS
  " Redo" 1 LABEL_KEY
}STATE_INIT

4 STATE{
  UF1 ?KEY
  ACTION{
    " Ready for initial alignment." W.NOTICE
    0 NEW_STATE
  }ACTION
}STATE

```

16.2 COO_CCITT.F

The COO_CCITT.F test script demonstrates the simple exchange of changeover messages. The simulation is started via a function key or upon receipt of a COO message. As the test script does not contain the logic to simulate all level 2 or level 3 functions associated with changeover and changeback, the script can only run on two simulation ports connected back to back. COO_ANSI.F is identical to this example except for message strings used in the STRING->BUFFER command.

```

( ----- )
( File Title: COO_CCITT.F )
( Version: 2.0 )
( This script simulates CHM or SNTM message exchange )
( ----- )

TCLR                ( initialize the test manager )

#IFNOTDEF COO_CCITT.F
0 CONSTANT COO_CCITT.F      ( signature flag to avoid recompiling definitions )

0 CONSTANT COO            ( CHM and SNTM Messages )
1 CONSTANT COA
2 CONSTANT CBD
3 CONSTANT CBA
4 CONSTANT SLTM
5 CONSTANT SLTA
0x12345678 VARIABLE ROUTING_LABEL ( default routing label setting )

#ENDIF

1 SEQ{ ( MSU_BUFFER --- )
  <FIB> <>@ DROP BIBT!      ( set BIB and BSN to be transmitted )
  <FSN> <>@ DROP BSNT!
  CRC SWAP SEND_BUFFER_SU
}SEQ

```

```

2 SEQ{      ( -- )
  CRC <SIB> <># SEND_LSSU
}SEQ

3 SEQ{      ( -- )                ( Swap incoming OPC/DPC )
  <OPC> <>@ DROP                    ( Retrieve OPC )
  5 RUN_SEQ                          ( Copy received OPC into outgoing DPC )
  <DPC> <>@ DROP                    ( Retrieve DPC )
  6 RUN_SEQ                          ( Copy received DPC into outgoing OPC )
  7 RUN_SEQ                          ( Copy label into SU buffers )
}SEQ

4 SEQ{      ( type -- )            ( OPC or DPC prompt, depending on type )
  prompt
  10 STR>#
  IF
    DUP 1 0x3FFF BETWEEN?
    IF
      SWAP RUN_SEQ                  ( Run the sequence indicated by 'type' )
      7 RUN_SEQ                    ( Copy label into SU buffers )
    ELSE
      2DROP
      " Value out of range" W.ERROR
    ENDIF
  ELSE
    DROP
    " Invalid decimal number" W.ERROR
  ENDIF
}SEQ

5 SEQ{      ( point_code -- )      ( Store into DPC region of label )
  DUP 0xFF AND 0x00 0 8 RUN_SEQ    ( bits 1-8 )
  8 >># 0xC0 1 8 RUN_SEQ          ( bits 9-14 )
}SEQ

6 SEQ{      ( point_code -- )      ( Store into OPC region of label )
  DUP 0x3 AND 6 <<# 0x3F 1 8 RUN_SEQ ( bits 1-2 )
  DUP 2 >># 0xFF AND 0x00 2 8 RUN_SEQ ( bits 3-10 )
  10 >># 0xF0 3 8 RUN_SEQ         ( bits 11-14 )
}SEQ

7 SEQ{      ( -- )                ( Copy label into SU buffers )
  SLTA 1 + COO
  DO
    ROUTING_LABEL I BUFFER 10 + 4 CMOVE
  LOOP
}SEQ

8 SEQ{      ( value\mask\offset -- ) ( Alter portion of routing label )
  ROUTING_LABEL + DUP >R C@        ( Get byte which is to be modified )
  AND OR                          ( Modify bits identified by mask )
  R> C!                            ( Replace byte )
}SEQ

```

```
YES AUTO_LI                                ( Automatic LI setting )

X" 0000123456781119" COO STRING->BUFFER    ( Setup COO Tx buffer  )
X" 0000123456782119" COA STRING->BUFFER    ( Setup COA Tx buffer  )
X" 0000123456785112" CBD STRING->BUFFER    ( Setup CBD Tx buffer  )
X" 0000123456786112" CBA STRING->BUFFER    ( Setup CBA Tx buffer  )
X" 00011234567811C054455854204D455353414745" SLTM STRING->BUFFER
X" 00011234567821C054455854204D455353414745" SLTA STRING->BUFFER

( ***** STATE MACHINE ***** )
0 STATE_INIT{
  NO AUTO_FISU                            ( Don't repeat FISUs )
  YES AUTO_FSN                             ( Automatic FSN setting )
  1 BIBT! 127 BSNT! 1 FIBT! 127 FSNT!    ( Reset BIB, BSN etc. to all one's )
  CLEAR_KEYS                              ( Clear all testkeys )
  " Start COO" 1 LABEL_KEY                ( Start change over sequence )
  " Start SLTM" 2 LABEL_KEY               ( Start test sequence )
  " Set OPC" 3 LABEL_KEY                  ( Set Originating Point Code )
  " Set DPC" 4 LABEL_KEY                  ( Set Destination Point Code )
}STATE_INIT

0 STATE{                                    ( Idle state )
  UF1 ?KEY                                ( Start change over sequence )
  ACTION{
    CRC COO SEND_BUFFER_SU
    1 NEW_STATE
  }ACTION

  UF2 ?KEY                                ( Start test sequence )
  ACTION{
    CRC SLTM SEND_BUFFER_SU
    1 NEW_STATE
  }ACTION

  UF3 ?KEY                                ( OPC prompt )
  ACTION{
    PROMPT" Enter Originating Point Code: "
    6 4 RUN_SEQ
    END_PROMPT
  }ACTION

  UF4 ?KEY                                ( DPC prompt )
  ACTION{
    PROMPT" Enter Destination Point Code: "
    5 4 RUN_SEQ
    END_PROMPT
  }ACTION
```

```

<COO> ?RX_FRAME
ACTION{
    3 RUN_SEQ                ( Set buffers' routing label )
    COA 1 RUN_SEQ            ( Send Change Over Ack )
    1 NEW_STATE
}ACTION

<SLTM> ?RX_FRAME
ACTION{
    3 RUN_SEQ                ( Set buffers' routing label )
    SLTA 1 RUN_SEQ          ( Send Test Ack )
    1 NEW_STATE
}ACTION
}STATE

1 STATE_INIT{
    CLEAR_KEYS " Stop" 1 LABEL_KEY
    YES AUTO_FISU
}STATE_INIT

1 STATE{
    ( Change over exchange state )
    <CHM> ?RX_FRAME
    ACTION{
        <H1> <>@ DROP DOCASE
            CASE <COO> <># [ COA ]    ( Send Change Over Ack )
            CASE <COA> <># [ CBD ]    ( Send Change Back Decl )
            CASE <CBD> <># [ CBA ]    ( Send Change Back Ack )
            CASE DUP           [ COO ] ( Send Change Over Order )
        ENDCASE
        1 RUN_SEQ
    }ACTION

    <SNTM> ?RX_FRAME
    ACTION{
        <H1> <>@ DROP DOCASE
            CASE <SLTM> <># [ SLTA ]  ( Send Test Ack )
            CASE DUP           [ SLTM ] ( Send Test Message )
        ENDCASE
        1 RUN_SEQ
    }ACTION

    <SIB> ?RX_FRAME          ( LSSU type SIB received? )
    ACTION{
        0 NEW_STATE
    }ACTION

    UF1 ?KEY                ( Stop key )
    ACTION{
        2 RUN_SEQ            ( Send LSSU type SIB )
        0 NEW_STATE
    }ACTION
}STATE

```

A

CODING CONVENTIONS

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.

A.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, n_1 and n_2 are numbers and the flag is either 0 for a false result, or 1 for a true result. This same example could also be written as follows.

```
( n1\n2 -- 0|1 )
```

The '\' character separates parameters when there is more than one. The parameters are listed from left to right with the leftmost item representing the bottom of the stack and the rightmost item representing the top of the stack.

The '|' character indicates that there is more than one possible output. The above example indicates that either a 0 or a 1 is returned on the stack after the '=' operation, with 0 being a false result, and 1 a true result.

The '[' and ']' characters surrounding a parameter(s) indicates that the parameter is not always present.

A.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.

Symbol	Description
a	Memory address
b	8 bit byte
c	7 bit ASCII character
n	16 bit signed integer
d	32 bit signed integer
u	32 bit unsigned integer
f	Boolean flag (0=false, non-zero=true)
ff	Boolean false flag (zero)
tf	Boolean true flag (non-zero)
s	String (actual address of a character string which is stored in a count prefixed manner)

Table A-1 ITL Symbols

A.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:

```
: HELLO ( -- )           ( Display text Hello in Notice Window )  
  " HELLO"              ( Create string )  
  W.NOTICE              ( 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.

A.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
```

A.5 Spacing and Indentation Guidelines

The following list outlines the general guidelines for spacing and indentations:

- One space between colon and name in colon definitions.
- One space between opening parenthesis and text in comments.
- One space between numbers and words within a definition.
- One space between initial " in strings (i.e. with " string", W." string", T." string", P." string", X" hex characters", etc...)
- One or more spaces at end of each line unless defining string which requires additional characters.
- Tab for nested constructs.
- Carriage return after colon definition and stack comment.
- Carriage return after last line of code in colon definition and semi-colon.

See the examples in Appendices A.6 and A.4.

A.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                ( Stack description )
  .....                      ( Comment for first line of code )
  IF
    .....                    ( Comment )
    DOCASE
      CASE X { ... }          ( Comment )
      CASE Y { ... }          ( Comment )
      CASE DUP { ... }        ( Comment )
    ENDCASE
  ELSE
    BEGIN
      .....                  ( Comment )
      .....                  ( Comment )
    UNTIL
  ENDIF
;
```

B

ASCII/EBCDIC/HEX CONVERSION TABLE

HEX	DEC	OCT	ASCII	EBCDIC	HEX	DEC	OCT	ASCII	EBCDIC
00	0	00	NUL	NUL	30	48	60	0	
01	1	01	SOH	SOH	31	49	61	1	
02	2	02	STX	STX	32	50	62	2	SYN
03	3	03	ETX	ETX	33	51	63	3	IR
04	4	04	EOT	PF	34	52	64	4	PP
05	5	05	ENQ	HT	35	53	65	5	TRN
06	6	06	ACK	LC	36	54	66	6	NBS
07	7	07	BEL	DEL	37	55	67	7	EOT
08	8	10	BS	GE	38	56	70	8	SBS
09	9	11	HT	SPS	39	57	71	9	IT
0A	10	12	LF	RPT	3A	58	72	:	RFF
0B	11	13	VT	VT	3B	59	73	;	CU3
0C	12	14	FF	FF	3C	60	74	<	DC4
0D	13	15	CR	CR	3D	61	75	=	NAK
0E	14	16	SO	SO	3E	62	76	>	
0F	15	17	SI	SI	3F	63	77	?	SUB
10	16	20	DLE	DLE	40	64	100	@	SP
11	17	21	DC1	DC1	41	65	101	A	
12	18	22	DC2	DC2	42	66	102	B	
13	19	23	DC3	DC3	43	67	103	C	
14	20	24	DC4	RES	44	68	104	D	
15	21	25	NAK	NL	45	69	105	E	
16	22	26	SYN	BS	46	70	106	F	
17	23	27	ETB	POC	47	71	107	G	
18	24	30	CAN	CAN	48	72	110	H	
19	25	31	EM	EM	49	73	111	I	
1A	26	32	SUB	UBS	4A	74	112	J	cent
1B	27	33	ESC	CUI	4B	75	113	K	.
1C	28	34	FS	IFS	4C	76	114	L	<
1D	29	35	GS	IGS	4D	77	115	M	(
1E	30	36	RS	IRS	4E	78	116	N	+
1F	31	37	US	IUS	4F	79	117	O	
20	32	40	SP	DS	50	80	120	P	&
21	33	41	!	SOS	51	81	121	Q	
22	34	42	"	FS	52	82	122	R	
23	35	43	#	WUS	53	83	123	S	
24	36	44	\$	BYP	54	84	124	T	
25	37	45	%	LF	55	85	125	U	
26	38	46	&	ETB	56	86	126	V	
27	39	47	'	ESC	57	87	127	W	
28	40	50	(SA	58	88	130	X	
29	41	51)	SFE	59	89	131	Y	
2A	42	52	*	SM/SW	5A	90	132	Z	!
2B	43	53	+	CSP	5B	91	133	[\$
2C	44	54	,	MFA	5C	92	134	\	
2D	45	55	-	ENQ	5D	93	135])
2E	46	56	.	ACK	5E	94	136	^	;
2F	47	57	/	BEL	5F	95	137	_	~

HEX	DEC	OCT	ASCII	EBCDIC	HEX	DEC	OCT	ASCII	EBCDIC
60	96	140	`	-	90	144	220		
61	97	141	a	/	91	145	221		j
62	98	142	b		92	146	222		k
63	99	143	c		93	147	223		l
64	100	144	d		94	148	224		m
65	101	145	e		95	149	225		n
66	102	146	f		96	150	226		o
67	103	147	g		97	151	227		p
68	104	150	h		98	152	230		q
69	105	151	i		99	153	231		r
6A	106	152	j		9A	154	232		
6B	107	153	k		9B	155	233		}
6C	108	154	l	·	9C	156	234		□
6D	109	155	m	%	9D	157	235)
6E	110	156	n	-	9E	158	236		+
6F	111	157	o	>	9F	159	237		■
70	112	160	p	?	A0	160	240		-
71	113	161	q	·	A1	161	241		o
72	114	162	r		A2	162	242		s
73	115	163	s		A3	163	243		t
74	116	164	t		A4	164	244		u
75	117	165	u		A5	165	245		v
76	118	166	v		A6	166	246		w
77	119	167	w		A7	167	247		x
78	120	170	x		A8	168	250		y
79	121	171	y	\	A9	169	251		z
7A	122	172	z	:	AA	170	252		
7B	123	173	{	#	AB	171	253		L
7C	124	174		@	AC	172	254		┌
7D	125	175	}	'	AD	173	255		[
7E	126	176	-	"	AE	174	256		└
7F	127	177	DEL	"	AF	175	257		•
80	128	200		"	B0	176	260		0
81	129	201		a	B1	177	261		1
82	130	202		b	B2	178	262		2
83	131	203		c	B3	179	263		3
84	132	204		d	B4	180	264		4
85	133	205		e	B5	181	265		5
86	134	206		f	B6	182	266		6
87	135	207		g	B7	183	267		7
88	136	210		h	B8	184	270		8
89	137	211		i	B9	185	271		9
8A	138	212			BA	186	272		
8B	139	213		{	BB	187	273		└
8C	140	214		<	BC	188	274		┐
8D	141	215		(BD	189	275]
8E	142	216		+	BE	190	276		*
8F	143	217		†	BF	191	277		-

HEX	DEC	OCT	ASCII	EBCDIC	HEX	DEC	OCT	ASCII	EBCDIC
C0	192	300		{	F0	240	360		0
C1	193	301		A	F1	241	361		1
C2	194	302		B	F2	242	362		2
C3	195	303		C	F4	244	364		4
C4	196	304		D	F3	243	363		3
C5	197	305		E	F5	245	365		5
C6	198	306		F	F6	246	366		6
C7	199	307		G	F7	247	367		7
C8	200	310		H	F8	248	370		8
C9	201	311		I	F9	249	371		9
CA	202	312			FA	250	372		
CB	203	313			FB	251	373		
CC	204	314			FC	252	374		
CD	205	315			FD	253	375		
CE	206	316			FE	254	376		
CF	207	317			FF	255	377		
D0	208	320		}					
D1	209	321		J					
D2	210	322		K					
D3	211	323		L					
D4	212	324		M					
D5	213	325		N					
D6	214	326		O					
D7	215	327		P					
D8	216	330		Q					
D9	217	331		R					
DA	218	332							
DB	219	333							
DC	220	334							
DD	221	335							
DE	222	336							
DF	223	337							
E0	224	340		\					
E1	225	341							
E2	226	342		S					
E3	227	343		T					
E4	228	344		U					
E5	229	345		V					
E6	230	346		W					
E7	231	347		X					
E8	232	350		Y					
E9	233	351		Z					
EA	234	352							
EB	235	353							
EC	236	354							
ED	237	355							
EE	238	356							
EF	239	357							

C**COMMAND CROSS REFERENCE LIST**

This appendix cross references old commands and tokens, 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.

Old Command	New Command
<token>@	<token> <>@
<token>#	<token> <>#

Old Token	New Token
<INT>	<NI=INT>
<INT0>	<NI=INS>
<NAT>	<NI=NAT>
<NAT0>	<NI=NAS>
<SIB2>	<LSS2.SIB>
<SIE2>	<LSS2.SIE>
<SIN2>	<LSS2.SIN>
<SIO2>	<LSS2.SIO>
<SIOS2>	<LSS2.SIOS>
<SIPO2>	<LSS2.SIPO>

D

SAMPLE CONFIGURATION FILE

The SS#7 application can automatically be configured by creating an appropriately named configuration file (eg. SS7_MON.D3 for PRA Monitor application 1). The configuration file is executed when the application is initially loaded. Refer to the Programmer's Reference Manual for valid configuration filenames.

The following program is a sample configuration file for SS#7.

```
( Protocol set selection )

" CCITT_88" SELECT_VAR      ( select the CCITT 1988 Blue Book Variance )

" CCITT_ISUP88.T"
" CCITT_NET88.T"
" CCITT_LINK88.T"
3 LOAD_PROTOCOL_SET        ( load the ISUP protocol, with supporting underlying
                           ( layers )

( Display Format )

TRACE_COMP                 ( set the trace mode format to complete )
TIME_ON                    ( display timestamps )
REP_COMP                   ( change display to complete mode )
<ISUP> #MSG_DETAIL_FORMAT  ( display ISUP detail in message format )

( Disk Recording )

DISK_WRAP                  ( set disk recording mode to overwrite )
RECORD WD1:DAT             ( start disk recording )

( Filters )

D_FILTER                   ( select the disk filter )
<FISU> F-                  ( block FISU's )
<LSSU> F-                  ( block 1 byte LSSU's )
<LSS2> F-                  ( block 2 byte LSSU's )
ACTIVATE_DISK              ( activate disk filter )
```

(Level 2 configuration)

5000 COMPRESS-SU! (set the SU compression ratio)

(Level 1 configuration)

ON-LINE (turn application on-line)

INDEX

- <>@, 7-4
- <>#, 7-3
- <: :>, 7-5
- Abort
 - test manager event, 15-4, 15-5
 - transmitting, 14-3
- ?ABORT, 15-5
- ACTION{ }ACTION, 15-1
- ACTIVATE_DISK, 9-1
- ACTIVATE_RAM, 9-1
- ACTIVATE_REPORT, 9-1
- ACTIVATE_TEST, 9-2
- AERM, *see* Initial Alignment Error Rate Monitor
- AERM-E-THRESHI, 13-6
- AERM-E-THRESH@, 13-6
- AERM-FUNCTIONI, 13-6
- AERM-FUNCTION@, 13-6
- AERM-N-THRESHI, 13-6
- AERM-N-THRESH@, 13-6
- <>@_ALL, 7-4
- ALLOT_BUFFER, 14-3
- APPEND->BUFFER, 14-2
- APPEND_TO_BUFFER, 14-3
- Architecture
 - monitor, 2-1 to 2-4
 - simulation, 12-1 to 12-4
- Arming Triggers, 10-1
- AUTO_FISU, 13-5
- AUTO_FSN, 14-1
- AUTO_LI, 14-2
- B, *see* BACKWARD
- BACKWARD, 2-3
- Backward Sequence Number
 - setting, 13-5, 14-1
- BB, *see* SCRNB_BACK
- BIBTI, 14-1
- BIBT@, 14-1
- Bit Rate
 - simulation, 13-2
- Block Number
 - decode, 7-2, 11-1
 - display format, 6-5
- BLOCK-COUNT, 7-2, 11-1
- BOF-TSTAMP!, 3-2, 13-3
- BOTH, 10-3
- BOTTOM, 2-4
- BSNT!, 14-1
- BSNT@, 14-1
- BUFFER, 14-3
- Buffer(s)
 - allocating memory, 14-3
 - appending, 14-2
 - appending text, 14-3
 - clearing, 14-3
 - moving text, 14-2, 14-3
 - numbers, 14-1
 - repeated transmission, 14-3, 14-4
 - structure, 14-1
 - transmitting, 14-3
- Capture RAM
 - capturing to RAM, 4-1
 - clearing, 4-2
 - configuring, 4-1
 - playback, 2-2 to 2-4
 - printing, 4-4
 - saving to disk, 4-2 to 4-4
 - trigger, 10-6
- CAPT_FULL, 4-2
- CAPT_FULL, 10-6
- +CAPT_FULL, 10-6
- CAPT_OFF, 4-1
- CAPT_ON, 4-1
- CAPT_WRAP, 4-1
- Character Set
 - ASCII, 6-5
 - EBCDIC, 6-5
 - hex, 6-5
 - JIS8, 6-6
- CLEAR_BUFFER, 14-3
- CLEAR_CAPT, 4-2
- CLEAR_KEYS, 15-2
- CLK-EXT_CLK, 13-2
- CLK=STD, 13-2
- Clocking
 - external, 13-2
 - standard, 13-2
- Comparison
 - anchored, 15-5
 - unanchored, 15-5
- COMPRESS-SUI, 3-3, 13-4
- COMPRESS-SU@, 3-3, 13-4
- Compression Ratio, 3-3, 13-4
- Configuration
 - capture RAM, 4-1
 - interface selection, 3-1
 - level 1, 3-1, 3-2, 13-1 to 13-3
 - level 2, 3-3, 3-4, 13-4 to 13-7
 - monitor, 3-1 to 3-5
 - restoring, 3-4, 13-8
 - saving, 3-4, 13-8
 - simulation, 13-1 to 13-9
- Connectors
 - V.11, 3-1, 13-2
 - V.28/RS-232C, 3-1, 13-2
 - V.35, 3-2, 13-2
 - V.36, 3-2, 13-2
- <COPIES>@, 7-2
- COUNTER1, 15-2
- CRC Error(s)
 - test manager event, 15-4, 15-5
 - transmitting, 14-3, 14-4
- ?CRC_ERROR, 15-5
- CS=ASCII, 6-5
- CS=EBCDIC, 6-5
- CS=HEX, 6-5
- CS=JIS8, 6-6
- CTOD_OFF, 4-3
- CTOD_ON, 4-3
- CTRACE, 9-5
- CURRENT_LSSU, 14-4
- C_FILTER, 9-2
- DEACTIVATE_DISK, 9-2
- DEACTIVATE_RAM, 9-1
- DEACTIVATE_REPORT, 9-1
- DEACTIVATE_TEST, 9-2
- Decoder
 - level 1, 11-1
 - signal unit, 7-1 to 7-5
- Destination Point Code, *see* Point Codes
- DETAIL_FORMAT, 6-3
- DISK_FULL, 5-1
- DISK_FULL, 10-6
- +DISK_FULL, 10-6
- DISK_OFF, 5-2
- DISK_WRAP, 5-1
- Display Format
 - character, 6-2
 - complete, 6-2
 - dual, 6-4
 - hex, 6-2
 - short, 6-1
 - split, 6-2
 - timestamp, 6-5
 - trace statements, 6-2, 6-5
- DIS_REC, 5-2
- DPC, *see* Point Codes
- DTRACE, 9-5
- D_FILTER, 9-2
- ENB_REC, 5-2
- END-TIME, 7-2, 11-1
- ?EOF_IND, 15-4

INDEX [continued]

- Event Recognition, 15-3 to 15-6
 - abort, 15-4, 15-5
 - CRC error, 15-4, 15-5
 - errors, 15-4
 - level 1, 15-3
 - level 2, 15-4
 - mail, 15-6
 - signal unit, 15-4
 - timers, 15-6
 - using tokens, 15-5
 - wildcard, 15-6
- EVENT-TYPE@, 15-6
- F, see FORWARD
- F*, 9-3
- F*PC, 9-4
- F+, 9-3
- F+ALL, 9-3
- F+ALL_OTHERS, 9-4
- F+PC, 9-4
- F-, 9-3
- F-ALL, 9-3
- F-ALL_OTHERS, 9-4
- F-PC, 9-4
- FF, see SCRNFWD
- FIBT!, 14-1
- FIBT@, 14-1
- FILE->BUFFER, 14-2
- Filename
 - disk recording, 2-3
 - playback, 2-3
- FILL_BUFFER, 14-3
- Filters
 - activate, 9-1
 - deactivate, 9-1
 - routing labels, 9-4
 - tokens, 9-3
 - type, 9-2
- FISU
 - compression, 3-3, 13-4
 - idling, 13-5
- FORWARD, 2-3
- Forward Sequence Number
 - automatic incrementing, 14-1
 - setting, 13-5, 14-1
- FREEZE, 2-4, 12-4
- FROM_CAPT, 2-2
- FROM_DISK, 2-2
- FSNT!, 14-1
- FSNT@, 14-1
- FULL, 6-4
- F_PC_VALUE, 9-4
- F_RESET, 9-4
- F_RESET_ALL, 9-4
- F_STRING, 9-3
- F_VALUE, 9-3
- HALT, 2-2, 12-3
- HEADER_FORMAT, 6-3
- Hex, see Display Format
- HFILE->BUFFER, 14-2
- HIGHLIGHT, 10-7
- HIGHLIGHT=BLUE, 10-7
- HIGHLIGHT=RED, 10-7
- Identifiers
 - level 1, 11-1, 15-3
 - level 2, 15-4
 - receiver port, 7-1
- IF=V11, 3-1, 13-2
- IF=V28, 3-1, 13-2
- IF=V35, 3-2, 13-2
- IF=V36, 3-2, 13-2
- Initial Alignment Error Rate Monitor
 - activating, 13-6
 - emergency threshold, 13-6
 - maximum number, 13-6
 - threshold, 13-6
- Interface
 - V.11, 3-1, 13-2
 - V.28/RS-232C, 3-1
 - v.28/RS-232C, 13-2
 - V.35, 3-2, 13-2
 - V.36, 3-2, 13-2
- ?KEY, 15-6
- L1-ID@, 11-1
- L1_EVENT, 15-3
- ?L1_EVENT, 15-3
- L2_EVENT, 15-4
- ?L2_EVENT, 15-4
- <>L@, 7-4
- LEAD_CHECK, 13-3
- LEAD_OFF, 13-3
- LEAD_ON, 13-3
- Level 1
 - configuration, 3-1, 3-2, 13-1 to 13-3
 - decoder, 11-1
 - identifiers, 11-1, 15-3
 - test manager events, 15-3
- Level 2
 - configuration, 3-3, 3-4, 13-4 to 13-7
 - identifiers, 15-4
 - simulation, 14-1 to 14-4
 - test manager events, 15-4
- Live Data
 - monitor, 2-1
 - simulation, 12-1
 - simultaneous playback, 2-4, 12-3
 - triggers, 10-2
- LOAD_ALL, 3-5, 13-9
- LOAD_CONFIG, 3-4, 13-8
- LOAD_LABELS, 8-2
- LOAD_PROTOCOL_SET, 3-5, 13-8
- LOAD_RETURN_STATE, 15-2
- LONG-INTERVAL!, 6-6
- LSSU
 - compression, 3-3, 13-4
 - transmitting, 14-4
- ?MAIL, 15-6
- =MASK, 10-5
- MAX-RETRY!, 13-6
- MAX-RETRY@, 13-6
- MONITOR, 2-2, 12-2
- Monitor
 - architecture, 2-1 to 2-4
 - configuration, 3-1 to 3-5
 - live data, 2-1
 - online/offline, 3-2
- NEW_STATE, 15-2
- NEW_TM, 15-2
- Octet Counting Mode, 3-4, 13-5
- OCTET-MODE!, 3-4, 13-5
- OCTET-MODE@, 3-4, 13-5
- <: OF :>, 7-5
- OFF-LINE, 3-2, 13-3
- ON-LINE, 3-2, 13-3
- OPC, see Point Codes
- Origination Point Code, see Point Codes
- OTHER_EVENT, 15-6
- OTHER_FORMAT, 6-3
- >PC, 8-1
- PC->, 8-1
- PE->T4, 13-7
- PLAYBACK, 2-3
- Playback
 - capture RAM, 2-2
 - control, 2-3, 2-4
 - disk recording, 2-2 to 2-4

INDEX [continued]

- Playback [continued]
 - monitor, 2-2
 - simulation, 12-2
 - simultaneous live data, 2-4, 12-3
 - triggers, 10-3
- =PLAYBACK, 10-3
- PN->T4, 13-7
- Point Codes, 8-1
- PORT-ID, 7-1
- Printing
 - capture RAM, 4-2 to 4-4
 - disk recording, 4-3
 - throughput graph, 6-6
- PRINT_OFF, 4-4
- PRINT_ON, 4-4
- PRINT_TPR, 6-6
- QUIT_TRA, 4-2
- REC-LENGTH, 7-1
- REC-POINTER, 7-1
- ?RECEIVED, 15-5
- RECORD, 5-2
- Recording
 - captured data, 4-3, 4-4
 - filename, 2-3
 - live data to disk, 5-2
 - multi-channel, 5-3
 - overwrite, 5-1
 - playback disk, 2-2 to 2-4
 - stop, 5-2
 - suspend, 5-2
- REP_CHAR, 6-2
- REP_COMP, 6-2
- REP_HEX, 6-2
- REP_OFF, 6-1
- REP_ON, 6-1
- REP_SHORT, 6-1
- REP_SPLIT, 6-2
- REP_TRACE, 6-2
- RESET_ROUTING_LABELS, 8-2
- Restoring
 - configurations, 3-4, 13-8
 - routing labels, 8-2
- RETURN_STATE, 15-2
- RL_HEX, 6-7
- RL_TEXT, 6-7
- Routing Labels, 8-1, 8-2
 - filters, 9-4
 - restoring, 8-2
 - saving, 8-2
 - subfields, 8-1
 - triggers, 10-5
- RTRACE, 9-5
- RUN_SEQ, 15-2
- =RX, 10-2
- ?RX_ERROR, 15-4
- ?RX_FRAME, 15-4
- ?RX_GOOD, 15-5
- R_FILTER, 9-2
- SAVE_CONFIG, 3-4, 13-8
- SAVE_LABELS, 8-2
- Saving
 - configurations, 3-4, 13-8
 - routing labels, 8-2
- SCALEI, 6-7
- Screen, scroll control, 2-3, 2-4
- SCRN_BACK, 2-3
- SCRN_FWD, 2-3
- ?SEARCH, 15-5
- SEE_TRA, 4-3
- SELECT_VAR, 3-5, 13-9
- SEND_BUFFER_SU, 14-3
- SEND_FISU, 14-4
- SEND_FLAGS, 14-4
- SEND_LSSU, 14-4
- SEND_MARKS, 14-4
- SEND_SPACES, 14-4
- SEQ{ }SEQ, 15-2
- SET_DFC, 8-1
- SET_OPC, 8-1
- SET_SLS, 8-2
- SF_SIZE, 14-4
- SHORT-INTERVALI, 6-6
- SIF, see Signalling Information Field
- SIF-MAXSIZEI, 3-4, 13-5
- SIF-MAXSIZE@, 3-4, 13-5
- Signal Unit(s)
 - decoder, 7-1 to 7-5
 - length, 7-1
 - length indicator, 14-2
 - test manager event, 15-4
- Signalling Information Field, 3-4, 13-5
- Signalling Link Selection, 8-2
- SIM=DCE, 13-1
- SIM=DTE, 13-1
- Simulation
 - architecture, 12-1 to 12-4
 - bit rate, 13-2
 - configuration, 13-1 to 13-9
 - level 2, 14-1 to 14-4
 - live data, 12-1
 - online/offline, 13-3
 - playback, 12-2
- SLS, see Signalling Link Selection
- SPEEDI, 13-2
- SPEED@, 3-2, 13-3
- SS7_TIMERI, 13-7
- SS7_TIMER@, 13-7
- SS7_TIMER_DEFAULT@, 13-7
- START-TIME, 7-2
- START_SS7_TIMER, 13-7
- State Machine, 15-1
- STATE_INIT{ }STATE_INIT, 15-1
- STATE{ }STATE, 15-1
- =STRING, 10-5
- STRING, 10-5
- +STRING, 10-5
- STRING->BUFFER, 14-2
- SUERM
 - activating, 3-3, 13-4
 - counter, 3-4, 13-5
 - threshold, 3-3, 13-5
- SUERM-COUNTI, 3-4, 13-5
- SUERM-COUNT@, 3-4, 13-5
- SUERM-FUNCTIONI, 3-3, 13-4
- SUERM-FUNCTION@, 3-3, 13-4
- SUERM-THRESHI, 3-3, 13-5
- SUERM-THRESH@, 3-3, 13-5
- T+, 10-3
- T+ALL, 10-3
- T+PC, 10-5
- T-, 10-3
- T-ALL, 10-3
- T-ALL_OTHERS, 10-4
- T-PC, 10-5
- ?T4=PE, 13-7
- T=ALL_OTHERS, 10-4
- TA1, 10-6
- TA2, 10-6
- TA3, 10-6
- TA4, 10-7
- TCLR, 15-1
- Test Manager
 - action definition, 15-1
 - event recognition, 15-3
 - execution, 15-2
 - initialization, 15-1
 - loading multiple scripts, 15-2
 - sequences, 15-2
 - state definition, 15-1
 - state initialization, 15-1
 - state transition, 15-2
 - stopping the, 15-2

INDEX [continued]

- Test Manager *[continued]*
 - subroutines, 15-2
- TF_RESET, 9-4, 10-4
- TF_RESET_ALL, 9-4, 10-4
- Throughput Graph
 - displaying, 6-6
 - long interval, 6-6
 - printing, 6-6
 - scaling, 6-7
 - short interval, 6-6
- TIME, 10-6
- +TIME, 10-6
- TIMEOUT, 15-6
- ?TIMER, 15-6
- Timers
 - duration, 13-7
 - setting, 13-7
 - starting, 13-7
 - test manager event, 15-6
- Timestamp
 - beginning of frame, 3-2, 13-3
 - decode, 7-2, 11-1
 - display format, 6-5
- TIME_DAY, 6-5
- TIME_OFF, 6-5
- TIME_ON, 6-5
- TITLE, 2-3
- TM_STOP, 15-2
- Tokens
 - decode, 7-4
 - filters, 9-3
 - test manager event, 15-5
 - triggers, 10-3
- TOP, 2-3
- TPR_OFF, 6-6
- TPR_ON, 6-6
- TR1, 10-2
- TR1_OFF, 10-1
- TR1_ON, 10-1
- TR2, 10-2
- TR2_OFF, 10-2
- TR2_ON, 10-1
- TR3, 10-2
- TR3_OFF, 10-2
- TR3_ON, 10-1
- TR4, 10-2
- TR4_OFF, 10-2
- TR4_ON, 10-1
- Trace Statements, 6-2
 - display format, 6-5
- TRACE_COMP, 6-5
- TRACE_SHORT, 6-5
- TRANSFER, 4-2
- Transmitting
 - aborts, 14-3
 - buffers, 14-3
 - CRC error, 14-3, 14-4
 - LSSU, 14-4
 - mark character, 14-4
 - space character, 14-4
- TRA_ALL, 4-2
- TRA_END, 4-3
- TRA_START, 4-2
- Triggers
 - actions, 10-6
 - arming, 10-1
 - capture RAM full, 10-6
 - conditions, 10-2 to 10-6
 - direction, 10-2
 - disk full, 10-6
 - highlighting, 10-7
 - live data, 10-2
 - playback data, 10-3
 - routing labels, 10-5
 - selection, 10-2
 - string, 10-5
 - string mask, 10-5
 - time, 10-6
 - tokens, 10-3
- TURN_OFF_LABEL_FILTERS, 9-5
- TURN_OFF_LABEL_TRIGGERS, 10-5
- TX, 10-3
- T_FILTER, 9-2
- T_PC_VALUE, 10-4
- T_RESET, 10-4
- T_RESET_ALL, 10-4
- T_STRING, 10-4
- T_VALUE, 10-4
- Wildcard(s)
 - comparison, 15-5
 - test manager event, 15-6