USM

PROGRAMMER'S MANUAL

IDACOM®
PROTOCOL TESTING SPECIALISTS

USM PROGRAMMER'S MANUAL

September 1990
Version 2.0

UNIVERSAL MONITOR/SIMULATION

Version 2.0

1.1 Enhancements

Display Format
The Baudot character set can now be selected for data display in ASYNC framing with 5 bits/character.

Configuration Menu
Mark and space parity settings have been added (maximum 7 data bits) for ASYNC and CHARACTER SYNC framing.

New configuration commands have been added for test script simplification for:
- Bit Rate;
- Sync Character;
- Message Length; and
- Message Timeout.

Refer to the Configuration section of the USM Programmer's Manual for more information.

Receive Data Lead Transitions
Commands have been added to recognize data lead changes for display, RAM capture, or disk. Data lead transitions must be requested in a test script before they can be detected. The received frame indications are not affected by data lead indications. See the USM Programmer's Manual for more information.

WARNING
All configuration changes must be done prior to requesting recognition of received data lead transitions.

NOTE
Received data lead transitions are reported as ON when the line remains in a high state (i.e. a steady space has been received).

Received data lead transitions are reported as OFF when the line remains in a low state (i.e. a steady mark has been received).

Transmit Data Lead Transitions (Simulation Only)
Commands have been added to set the transmit data lead high or low. The line remains in the set state until the next TXD_ON, TXD_FF, or send data command. Refer to the USM Programmer's Manual for details.

Data Leads
Received data leads (available in release 1.4) are now captured to RAM and data recordings for later playback.

- **Message Length**
  Message length in CHARACTER SYNC and ASYNC can now be disabled. Previously in CHARACTER SYNC, this was referred to as unlimited.

- **Message Timeout**
  Message timeout in ASYNC can now be disabled.

- **End of Frame Character**
  End of Frame Character has been added to the Configuration Menu (valid in ASYNC only). Up to 4 separate characters can be defined to terminate the end of a received data block. Refer to the USM Programmer's Manual for the corresponding commands.

- **Saving Configurations**
  The interface type and end of frame characters are now saved in the specified configuration file created using the *Save Config* function key.

- The MAKE_DATA1 through MAKE_DATA8 commands are now available in the monitor.

1.2 Changes

- **Variables**
  The BYTE-TIME variable is no longer available. Contact IDACOM/HP customer support if your test script requires this variable.

  The START-TIME variable must be used instead of the T/RXD-TIME variable (Version 1.4) for received data lead transitions in test scripts. T/RXD-TIME still must be used for the timestamp of transmitted data lead indications.

- **Setting Message Length in ASYNC**
  Any test scripts written using the EOF_COUNT command in the ASYNC protocol must be modified.

  Example:
  Set the message length to 1 character.

  For versions prior to 2.0:
  ```
  PORT @ 0 EOF_COUNT
  CHANGE_CONFIG
  ```

  For this and subsequent versions:
  ```
  1=EOF_COUNT                ( Preferred method )
  ```

  or

  ```
  PORT @ 1 EOF_COUNT
  CHANGE_CONFIG
  ```

Reset Enable
Rest Idle on the Configuration Menu has been changed to *Reset Enable* to reflect correct functionality.

Interframe Fill
Interframe fill cannot be selected in Character SYNC, BISYNC EBCDIC, BISYNC ASCII, or ASYNC.

1.3 Problems Fixed

Defining Strings
The MAKE_DATA1 through MAKE_DATA8 commands no longer overwrite the passed string and now work with parity settings of mark and space.

Error Reporting
ASYNC parity errors are now reported as parity errors rather than BCC errors.

BISYNC abort errors are now reported as "ENQ in text" errors.

Trigger Actions
The trigger action of opening a disk recording no longer locks the tester when a disk error occurs. All triggers are now disarmed and an error message is displayed.

Simulation Only
The following notice is displayed when parity is set to none, ASYNC is chosen, and SEND_WITH_ERROR is called: 'String sent without parity error. Parity is set to none.'

When no characters are entered while constructing String1, the following message is displayed: 'String1 has not been entered'. Similar messages are displayed for String2, String3, and String4.

1.4 Errata

Page 10-3, USM Programmer's Manual
The following command should be worded as follows:

RXD_TRANS (-- address)
Contains the direction of the last received data lead transition. Possible values are N_TRANS (high state, a steady space has been received) or P_TRANS (low state, a steady mark has been received).

The following two commands have been added for data transmission control:

WAIT_ON (--)
Queues a frame/block for transmission and pauses the application until the entire frame/block is transmitted.

NOTE
Use WAIT_ON whenever leads are for flow control.

NOTE
The TO DTE Simulator with DCD Control set to ON, automatically queues a frame/block for transmission and pauses until the entire frame/block is transmitted.

WAIT_OFF (--)
Queues a frame/block for transmission and continues the application.

PREFACE

This manual is intended to provide a programmer's guide to the Universal Simulation/Monitor programs, hereafter referred to as USM. General programming information is provided in the Programmer's Reference Manual. Information contained in this manual is machine independent.

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

P/N 6000–1202

IDACOM Electronics Ltd.
A division of Hewlett–Packard

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

TABLE OF CONTENTS

TABLE OF CONTENTS [continued]

TABLE OF CONTENTS [continued]

LIST OF FIGURES

LIST OF TABLES

1
INTRODUCTION

USM supports monitoring and testing of most internationally used synchronous and asynchronous data communication protocols. These include bit oriented protocols such as HDLC, SDLC, X.25, SNA, Teletex, Fax Group IV, and X.75; character oriented protocols such as Bisync ASCII, Bisync EBCDIC, and Async data. Only layer 1 information is decoded; no automatic protocol decoding is performed but can be implemented in user-written test scripts. Data is displayed in character or hex format. Triggers, RAM capture, disk recording, and some filters are provided. An autoconfiguration feature is available in the monitor.

The simulation provides responses to received events through user-written test scripts. For built-in automatic responses to received events, the appropriate IDACOM emulation application should be used (eg. X.25 Emulation).

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.

```
              Input              Output
            Parameters         Parameters

          ( Par1 \ Par2  --  Par3 \ Par4 \ Par5 )
                      ↑    ↑                 ↑
Item on top of stack       |          Item on top of stack
       Input/Output Separator
```

Figure 1-1 Sample Stack Comment

NOTE
See Appendix C for further explanation of stack parameters.

Sample complete test scripts are supplied in Section 11. These test scripts are also supplied on disk with the application program.

The USM application can be controlled remotely from a terminal. All commands described in this manual can be entered from a remote terminal's keyboard followed by a ↵ (RETURN). The application processes the remote command and returns the 'ROK' prompt to the remote terminal. The remote terminal must be connected to the modem port on the back of the tester. To configure the application for remote control, refer to the Programmer's Reference Manual.

2
CONFIGURATION

Simulation and monitor configuration is identical with two exceptions:

- Autoconfiguration is not available in the simulation
- Simulation mode is not available in the monitor

WARNING
The Universal Simulation/Monitor should be in offline mode when making configuration changes to prevent reception of invalid data or problems on the line.

GO_ONLINE (--)
Turns the interface data and lead receivers on, and returns the simulation to the selected simulation mode.

Online function key (highlighted)

GO_OFFLINE (--)
Turns the interface data and lead receivers off (default). The simulation goes into passive monitor mode.

Online function key (not highlighted)

```
                      Simulation Configuration Menu

 → Interface Type    RS232C/V.28        Interface Leads          DISABLED
   Simulation Mode   TO DCE

  Protocol Configuration:
   Framing           HDLC/SDLC          Reset Enable             ---
   Clocking          NRZ WITH CLOCK     Sync Reset Character     ---
   Bit Rate          64000              DCD Control              OFF
   Bits/Character    8                  CRC                      CCITT
   Stop Bits         ---                Strip Sync               ON
   Parity            NONE               Message Length           ---
   Sync Character    HEX 7E             Message Timeout          ---
   Interframe Fill   SYNC               End of Frame Character   ---
```

Figure 2-1 Simulation Configuration Menu

2.1 Interface Type

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

RS232C/V.28 function key

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

RS422/V.11 function key

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

V.35 function key

IF=V36 (--)
Selects the V.36/RS-449 connector and electrically isolates the other connectors on the port.

RS449/V.36 function key

NOTE
A WAN tester has a V.28, V.11, and either a V.35 or *V.36 connector. These commands are only applicable if the program is running on a WAN interface.*

2.2 Simulation Mode

Selects the physical type of simulation and determines whether the tester generates or expects to receive clocking, as well as setting which pins transmit and receive data.

=SIM_DTE (--)
Selects the 'to DTE' interface. Clocking must be supplied by the attached equipment.

TO DTE function key

=SIM_DCE (--)
Selects the 'to DCE' interface. The tester supplies all necessary clocking information to the interface connector.

TO DCE function key

NOTE
When the simulation is running on a B-Channel, only the TO DCE interface is allowed. Thus, the command =SIM_DCE is ignored.

2.3 Interface Leads

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

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

Example:
Enable the request to send lead.
```
IRS ENABLE_LEAD
```

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

Example:
Disable the clear to send lead.
```
ICS DISABLE_LEAD
```

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

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

ENABLED function key

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

DISABLED function key

2.4 Protocol Configuration

→ *Framing*

WARNING
Framing must be the first item selected. All other items, except bit rate, change to the default configuration for each framing type. See Appendix A for framing formats.

P=BOP[HDLC/SDLC] (--)
Selects bit–oriented procedure (default) with the following defaults:

- NRZ clocking
- 8 bits per character
- No parity
- Sync character of hex 7E
- Interframe fill character is the sync character
- DCD control is off
- CRC calculation according to CCITT
- Strip sync is on
- ASCII character set

HDLC/SDLC function key

P=COP_SYNC (--)
Selects character–oriented procedure with the following defaults:

- NRZ clocking
- 8 bits per character
- No parity
- Sync character of hex 16
- Interframe fill is marking
- Reset enable is on
- Sync reset character of hex FF
- DCD control is off
- No CRC calculation
- Strip sync is on
- Message or block length is disabled
- ASCII character set

CHARACTER SYNC function key

P=EBCDIC_BISYNC (--)

Selects Bisync EBCDIC framing with the following defaults:

- NRZ clocking
- 8 bits per character
- No parity
- Sync character of hex 32
- Interframe fill is marking
- DCD control is off
- CRC calculation according to CRC–16
- Strip sync is on
- EBCDIC character set

BISYNC EBCDIC function key

P=ASCII_BISYNC (--)

Selects Bisync ASCII framing with the following defaults:

- NRZ clocking
- 7 bits per character
- Odd parity
- Sync character of hex 16
- Interframe fill is marking
- DCD control is off
- CRC calculation according to VRC/LRC
- Strip sync is on
- ASCII character set

BISYNC ASCII function key

P=ASYNC (--)

Selects asynchronous framing with the following defaults:

- 8 bits per character
- 1 stop bit
- No parity
- DCD control off
- Message or block length is limited to 60 characters
- Timeout when 17 milliseconds occur between characters
- End of frame character is disabled
- ASCII character set

ASYNC function key

NOTE

P=ASYNC is ignored on the ISDN interfaces.

→ *Clocking*
IDACOM testers support four different clocking modes on a WAN interface. See Table A-1 for clocking modes and Figure A-6 for NRZ and NRZI data encoding.

CLK=STD (--)
Selects NRZ (non-return to zero) encoding with modem provided clocks (valid for all framing methods excluding ASYNC).

NRZ WITH CLOCK function key

CLK=EXT_CLK (--)
Selects a DTE provided transmit clock on pint 24 of an RS-232C connector (valid for all framing methods excluding ASYNC).

EXTERNAL TX CLOCK function key

CLK=NRZI (--)
Selects the non-return to zero inverted method of encoding with timing information extracted from the data signal (valid for HDLC/SDLC framing only).

NRZI function key

CLK=NRZIC (--)
Selects the non-return to zero inverted method of encoding with timing information extracted from the provided clock signal (valid for HDLC/SDLC framing only).

NRZI WITH CLOCK function key

→ *Bit Rate*
Monitor:
When asynchronous framing or NRZI clocking is selected, the interface speed must be selected from preset values on the Interface Port Speed Menu or set to a user-defined speed.

When synchronous framing and any other clocking mode is selected, the interface speed is measured, in bits per second, directly from the physical line.

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

NOTE
When asynchronous framing or a 'to DTE' interface is selected, the interface speed can only be selected from preset values on the Interface Port Speed Menu or set to a user-defined speed.

	TO DCE			
Clocking	**HDLC/SDLC**	**CHARACTER SYNC**	**BISYNC EBCDIC**	**BISYNC ASCII**
NRZ WITH CLOCK	Measure	Measure	Measure	Measure
EXTERNAL TX CLOCK	Select	Select	Select	Select
NRZI	Select	---	---	---
NRZI WITH CLOCK	Measure	---	---	---

Effect of Clocking and Simulation Mode Selections on Bit Rate

NOTE
Clocking is provided by the attached equipment when the bit rate can be selected.

=SPEED (bit rate--)
Specifies the number of bits per second and is used by the monitor to calculate throughput measurements. The port identifier can be obtained from the contents of the PORT variable.

Example:
Set the interface speed to 1200.

```
1200 =SPEED           ( Set the bit rate )
```

NOTE
The only interface speed allowed when the application is running on a B–Channel is 64000 bps.

→ *Bits/Character*
Selects the number of bits per character.

BITS/CHAR=8 (--)
Selects 8 bits per character (valid in HDLC/SDLC, CHARACTER SYNC, BISYNC EBCDIC, and ASYNC).

8 function key

BITS/CHAR=7 (--)
Selects 7 bits per character (valid in BISYNC ASCII, CHARACTER SYNC, and ASYNC).

7 function key

BITS/CHAR=6 (--)
Selects 6 bits per character (valid in CHARACTER SYNC and ASYNC).

6 function key

BITS/CHAR=5 (--)
Selects 5 bits per character (valid in CHARACTER SYNC and ASYNC).

5 function key

→ *Stop Bits*
Selects the number of stop bits per character (valid in ASYNC).

STOP_BITS=1.0 (--)
Selects 1 stop bit per character.

1 function key

STOP_BITS=1.5 (--)
Selects 1.5 stop bits per character.

1.5 function key

STOP_BITS=2.0 (--)
Selects 2 stop bits per character.

2 function key

→ *Parity*
Selects the checking method for character integrity during transmission. The parity is set during transmission and checked on reception.

PARITY=NONE (--)
Character integrity is not checked (valid in HDLC/SDLC, CHARACTER SYNC, BISYNC EBCDIC, and ASYNC).

NONE function key

PARITY=ODD (--)
Uses odd parity for checking character integrity (valid in CHARACTER SYNC, BISYNC ASCII, and ASYNC).

ODD function key

PARITY=EVEN (--)
Uses even parity for checking character integrity (valid in CHARACTER SYNC and ASYNC).

EVEN function key

PARITY=MARK (--)
Uses mark parity (parity bit is always equal to 1)), for checking character integrity (valid for CHARACTER SYNC and ASYNC).

MARK function key

PARITY=SPACE (--)
Uses space parity (parity bit is always equal to 0) for checking character integrity (valid for CHARACTER SYNC and ASYNC).

NOTE
Mark, space, and odd or even parity are not available when 8 bits per character is selected.

→ *Sync Character*
Selects the bit pattern which identifies the start and end of a block of data (not applicable in ASYNC).

SYNC=7E (--)
Sets the sync character to hex 7E (valid in HDLC/SDLC).

HEX 7E function key

SYNC=16 (--)
Sets the sync character to hex 16 (valid in BISYNC ASCII and CHARACTER SYNC).

HEX 16 function key

SYNC=32 (--)
Sets the sync character to hex 32 (valid in BISYNC EBCDIC and CHARACTER SYNC).

HEX 32 function key

SYNC=96 (--)
Sets the sync character to hex 96 (valid in CHARACTER SYNC).

HEX 96 function key

=SYNC (sync character --)
Specifies the sync character. Valid values for sync character are hex 0 through FF (valid in CHARACTER SYNC).

Example:
Set the sync character to hex FF.

```
0xFF =SYNC          ( Set the sync character )
```

SYNC function key

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

IF_FILL=SPACE (--)
Transmits the space bit pattern (all 0's) between blocks of data (valid in ASYNC).

IF_FILL=MARK (--)
Transmits the mark bit pattern (all 1's) between blocks of data (valid in all framing methods).

MARK function key

IF_FILL=SYNC (--)
Transmits sync characters between blocks of data (valid in HDLC).

SYNC function key

→ *Reset Enable*
Selects whether the sync reset character is enabled (valid in CHARACTER SYNC).

RESET_ENABLE_ON (--)
Enables the sync reset character.

ON function key

RESET_ENABLE_OFF (--)
Disables the sync reset character.

OFF function key

→ *Sync Reset Character*
Sets the character which causes the receiver to start a new sync search (valid in CHARACTER SYNC).

SYNC_RESET=FF (--)
Sets the sync reset character to hex FF (default).

HEX FF function key

=RESET (sync reset character--)
Specifies the sync reset character. Valid values are hex 0 through FF.

Example:
Set the sync reset character to hex 16.

```
0x16 =RESET           ( Define the sync reset character )
```

Modify Sync Reset function key

→ *DCD Control*
DCD_ON (--)
Turns on DCD control. The carrier detect lead must be on to receive data (valid in all but ASYNC).

ON function key

NOTE
The Universal Simulation 'to DTE' Simulation mode automatically turns on the carrier detect lead prior to transmitting data, and off after transmitting (when DCD control is turned on).

DCD_OFF (--)
Turns off DCD control (default). The state of the carrier detect lead does not affect data reception (valid in all framing methods).

OFF function key

→ *CRC*

CRC=CCITT (--)

Uses the CCITT Recommendation method for determining errors. A calculation is performed by the transmitter and a sixteen bit field (FCS) is attached to the end of the frame. The receiver performs the same calculation and the results should match those in the transmitted FCS bytes (valid in HDLC/SDLC).

CCITT function key

CRC=NONE (--)

The received frame is not checked for errors (valid in CHARACTER SYNC).

NONE function key

CRC=CRC_16 (--)

Uses the IBM BISYNC EBCDIC method for determining errors. A calculation is performed by the transmitter and a 16 bit field or BCC (block check character) is attached to the transmission block. The receiver performs the same calculation and the results should match those in the transmitted BCC (valid in EBCDIC BISYNC).

CRC–16 function key

CRC=VRC/LRC (--)

Uses the IBM BISYNC EBCDIC method for determining errors. VRC (vertical redundancy checking) is used to check each character as it is received. LRC (longitudinal redundancy checking) is used to check the entire block of data. The LRC character is calculated by the transmitting station and inserted at the end of the block as the BCC (valid in BISYNC ASCII).

VRC/LRC function key

→ *Strip Sync*

Selects whether SYNC characters are stripped by the receiver.

STRIP_SYNC_ON (--)

Strips sync characters (valid in all but ASYNC).

ON function key

STRIP_SYNC_OFF (--)

Sync characters are not stripped (valid in CHARACTER SYNC).

OFF function key

→ *Message Length*
Determines the length of a received data block (valid in ASYNC or CHARACTER SYNC).

NO_EOF_COUNT (--)
Character count is not used to determine the length of the received data block.

Example:
Turn off end of frame character count in ASYNC.

```
P=ASYNC                      ( Specify ASYNC )
NO_EOF_COUNT                 ( Turn off end of frame character count )
```

DISABLED function key

=EOF_COUNT (# of characters --)
Specifies the number of characters received before terminating a received data block.

Example:
Set the message length to 400 characters in ASYNC.

```
P=ASYNC                ( Specify ASYNC )
400 =EOF_COUNT         ( Specify 400 characters )
```

Modify Message Length function key

→ *Message Timeout*
ASYNC_TIME (milliseconds --)
Specifies the maximum elapsed time between characters before terminating a received data block (valid in ASYNC). Valid values are 1 through 65535 milliseconds.

Example:
Set the message timeout to 1000 milliseconds.

```
1000 ASYNC_TIME              ( Set the timeout )
```

Modify function key

NO_ASYNC_TIME (--)
Elapsed time between characters is not used to terminate a received data block (valid in ASYNC).

DISABLED function key

→ *End of Frame Character*
ENABLE_EOF_CHAR (character -- flag)
Enables a specified character used to terminate a received data block in ASYNC. Up to four different characters can be specified with values of hex 00 through FF. A true flag (1) is returned if successful, and a false flag (0) if an invalid character value or more than four characters have been enabled.

Specify Character function key
ENABLED function key

DISABLE_EOF_CHAR (character -- flag)
Disables a specified character used to terminate a received data block in ASYNC.

DISABLED function key

Example:
Specify and enable the first end of frame character as a carriage return (hex 0D).

```
0X0D 1 ASSIGN_EOF_CHAR       ( Specify character )
1 ENABLE_EOF_CHAR            ( Enable )
```

CLEAR_EOF_CHAR (--)
Disables all characters used to terminate a received data block in ASYNC (default).

2.5 Autoconfiguration

Autoconfiguration can be used when the line being monitored on a WAN interface has an unknown protocol to determine whether the protocol is bit-oriented (HDLC/SDLC), character-oriented (COP), BISYNC, or ASYNC. The characteristics are determined as shown in Table 2-1.

AUTO_CONF (--)
Automatically configures protocol parameters from the received data.

Monitor topic
Autoconfigure function key

Type of Protocol	Characteristics Determined
HDLC/SDLC	Baud Rate Encoding scheme (NRZ or NRZIC)
Character SYNC	Baud Rate SYNC Character (0x16, 0x32, 0x96)
BISYNC	Baud Rate Character Set (ASCII, EBCDIC)
ASYNC	Baud Rate Bits/Character (5, 6, 7, 8) Parity (NONE, ODD, EVEN)

Table 2–1 Autoconfiguration Parameters

Recognized baud rates for synchronous framing are 300, 1200, 2400, 4800, 7200, 9600, 14400, 16000, 19200, 38400, 56000, and 64000.

Recognized baud rates for asynchronous framing are 300, 1200, 2400, 4800, 7200, 9600, 14400, and 19200.

NOTE
If the line has a baud rate other than those listed previously, autoconfigure selects the closest supported speed.

During autoconfiguration, notices appear indicating the progress of the procedure. If autoconfiguration is successful, the monitor goes online and received data is displayed on the screen and captured to RAM; if autoconfiguration is unsuccessful, the following notice is displayed:

```
Configuration not found.
```

Framing	Autoconfiguration Time		
	300 bps	200–2400 bps	4800–64000 bps
SYNC	30 sec.	15 sec.	12 sec.
ASYNC	25 sec.	20 sec.	12 sec.

Table 2–2 Autoconfiguration Times

Autoconfiguration might fail to determine the configuration if the data circuit:

- is idle;
- contains small bursts of data;
- uses space for interframe fill or space for rest idle;
- contains synchronous data and the DCE clock line is not a one times (1x) clock; or
- carries a non–supported protocol.

3
MONITOR ARCHITECTURE

The Universal Monitor program monitors live data, saves data to capture RAM or disk, and displays data in a number of different formats. Triggers can perform specific actions when a specified event occurs.

3.1 Live Data

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

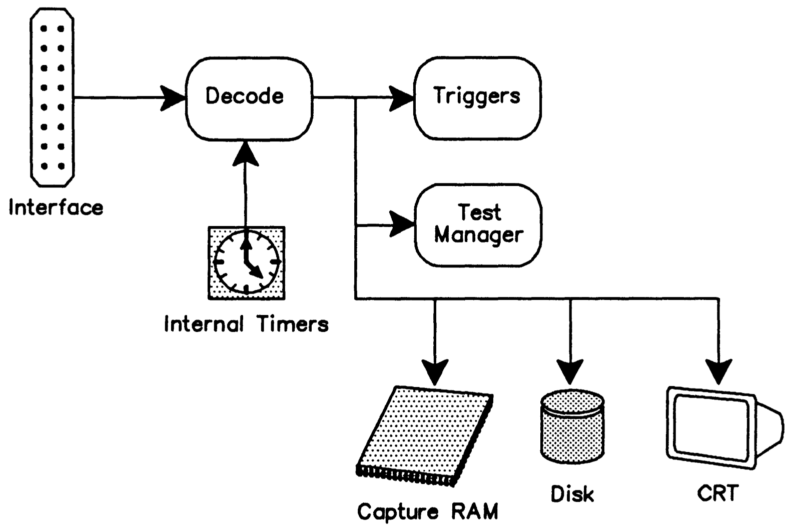


Figure 3–1 Universal Monitor Data Flow Diagram – Live Data

By default, the Universal Monitor 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.

3.2 Playback

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

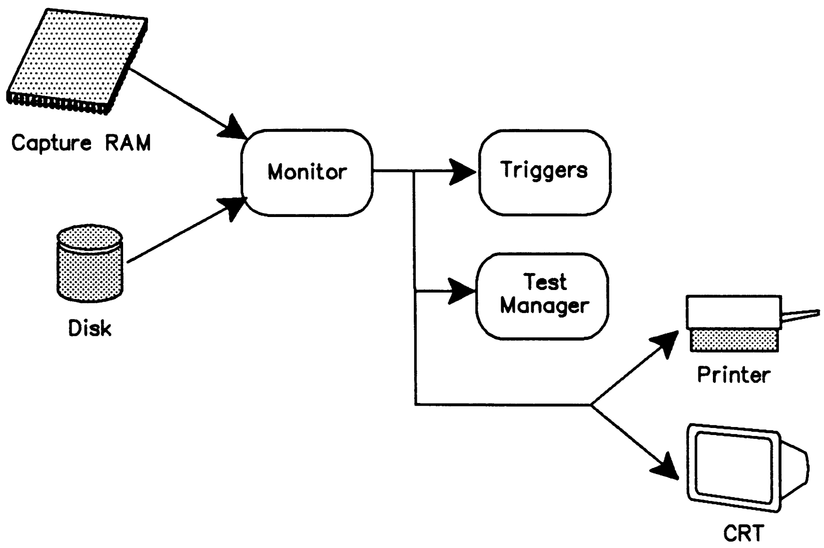


Figure 3–2 Universal Monitor Data Flow Diagram – Offline Processing

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 data from disk.
```
FROM_DISK              ( Identify a disk file as data source )
HALT                   ( Place the monitor in playback mode )
" ASYNC.1" =TITLE      ( Create title for next data file to be opened )
PLAYBACK               ( Playback data )
```

Playback Control

The following commands control display scrolling.

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

⇓ (Down 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 beginning of the playback source.

CTRL SHIFT ⇑

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

CTRL SHIFT ⇓

3.3 Simultaneous Live Data and Playback

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

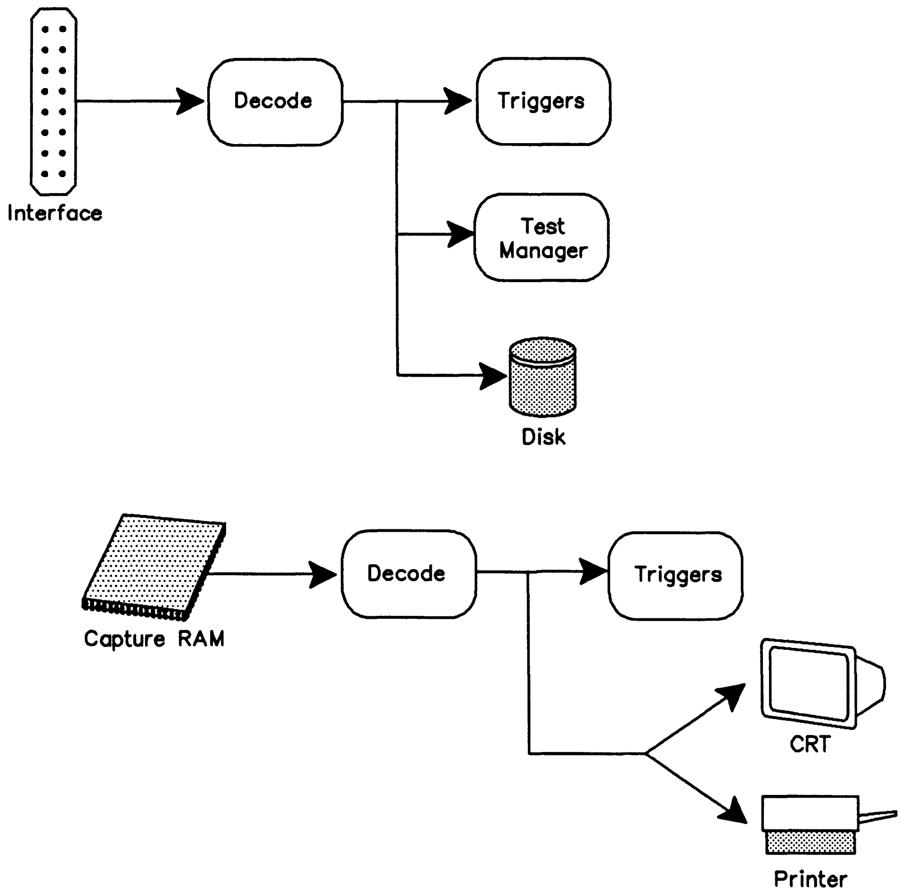


Figure 3–3 Universal Monitor Data Flow Diagram – Freeze Mode

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

FREEZE (--)
Enables data to be recorded to disk while data from capture RAM is played back.

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 3.2. Data stored in capture RAM can be transferred to disk.

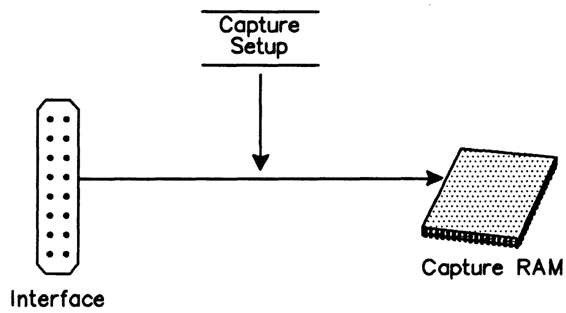


Figure 4–1 Universal Data Flow Diagram – Capture to RAM

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

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

Capture topic
Save RAM to Disk function key (highlighted)

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

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

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

Capture topic
Save RAM to Disk function key
All function key

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

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

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

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

SEE_TRA (--)
Displays the 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 'DATA1' and transfer all data from capture RAM to disk. After the transfer is complete, turn off data recording.

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

To Disk

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

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

To Printer

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

Print topic
Print On function key

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

Print topic
Print Off function key

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

```
FROM_CAPT                               ( 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 3.2. Data stored in capture RAM can be transferred to disk as described in Section 4.2.

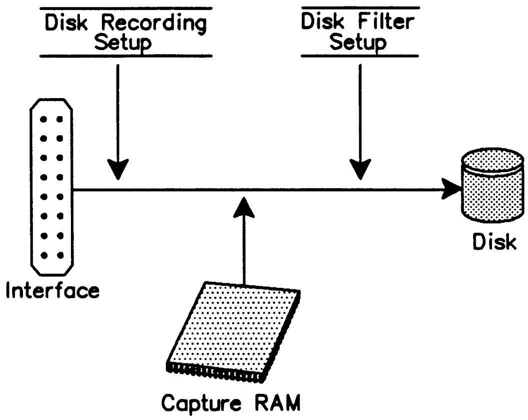


Figure 5–1 Universal Data Flow Diagram – Recording to Disk

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

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

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

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

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

RECORD (--)

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

Example:

```
RECORD DATA1
```

Capture topic
Record to Disk function key (highlighted)

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

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

Example:

```
Recording Start : Universal Mon     WAN RS232-C
V1.3-1.3 Rev 0                      PT500 - 24     SN# 03-1

Recording End   : Universal Sim     WAN RS232-C
V1.3-1.3 Rev 0                      PT500 - 24     SN# 03-1
```

DISK_OFF (--)

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

Capture topic
Record to Disk function key (not highlighted)

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

DIS_REC (--)

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

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

ENB_REC (--)

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

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

6
DISPLAY FORMAT

The Universal Monitor and Simulation applications can display data from the line (live data), from capture RAM, or from a disk recording in the following display formats:

- Hexadecimal
- Character
- Short
- Split
- Trace Statements

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

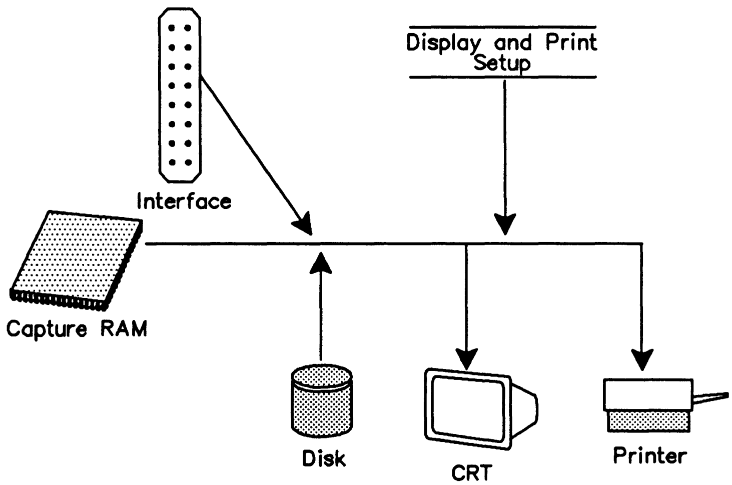


Figure 6–1 Universal Data Flow Diagram – Display and Print

NOTE
Data can only be printed in playback mode.

```
                         Display Format Menu

 → Display Format SHORT          Dual Window                OFF

   Timestamp      OFF            Trace Display Format       SHORT
   Character Set  ASCII
                                 Throughput Graph           OFF
                                   Short Interval (sec)     10
                                   Long  Interval (sec)     600
```

Figure 6–2 Display Format Menu

→ Display Format

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 condensed report (default). This includes the port identifier or timestamp, the length, and the first ten characters of data. This format is useful for higher speed monitoring as more frames per screen are displayed and processing is kept to a minimum.

SHORT function key

REP_HEX (--)

Displays timestamps or block sequence numbers and the port identifier in text. Frame contents are displayed in hex.

HEX function key

REP_CHAR (--)

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

CHARACTER function key

REP_NONE (--)
Displays only trace statements.

TRACE function key

SPLIT_ON (--)
Displays data in short format with a split screen display. The screen is divided in half with frames received from the DCE interface displayed on the left (Rx) and frames received from the DTE interface on the right (Tx).

SPLIT function key

SPLIT_OFF (--)
Sets the data display to the full screen short format display (default).

SHORT function key

REP_NONE (--)
Displays only trace statements.

TRACE function key

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

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

OFF function key

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

MM:SS.ssss function key

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

DD HH:MM:SS function key

→ *Character Set*
Selects the character set for data display.

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

ASCII function key

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

EBCDIC function key

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

HEX function key

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

TELETEX function key

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

JIS8 function key

R=BAUDOT (--)
Sets the character set for data display to Baudot (available in ASYNC framing with 5 bits/character).

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

Display topic
Clear function key

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

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

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

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 6–1 Dual Window Commands

→ *Trace Display Format*
Selects the display format for trace statements.

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

SHORT function key

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

COMPLETE function key

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

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

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

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

Example:
Set the throughput measurement speed to 2400.
```
2400 INTERFACE-SPEED !
TPR_ON
```

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

DISPLAY function key

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

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

OFF function key

PRINT_TPR (--)

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

DISPLAY AND PRINT function key

→ *Short Interval*

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

SHORT–INTERVAL (-- address)

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

Example:
Set the short interval to 20 seconds.

```
20 SHORT-INTERVAL !
TPR_ON
```

Modify Short Interval function key

→ *Long Interval*

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

LONG–INTERVAL (-- address)

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

Example:
Set the long interval to 300 seconds.

```
300  LONG-INTERVAL !
TPR_ON
```

Modify Long Interval function key

7
FILTERS

Filters provide the capability of passing or blocking specific events from the display, capture RAM, or disk recording. These three sets of filters act independently. This section describes the commands used to pass or block trace statements and lead changes.

```
            Filter Setup Menu

     Filter Type          DISPLAY
     Trace Statements     ON
   → Lead Changes         BLOCK
```

Figure 7–1 Filter Setup Menu

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

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

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

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

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

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

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

→ *Filter Type*
RAM function key
→ *Trace Statements*
ON function key

NO CTRACE (--)
Blocks trace statements from capture RAM.

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

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

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

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

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

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

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

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

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

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

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

→ *Filter Type*
RAM function key
→ *Lead Changes*
PASS function key

C1=NONE (--)
Blocks lead changes from capture RAM.

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

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

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

D1=NONE (--)
Blocks lead changes from disk.

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

8
DECODE

This section describes the data flow diagram for decoding, and lists the variables in which decoded information is saved. Only layer 1 decoding is performed.

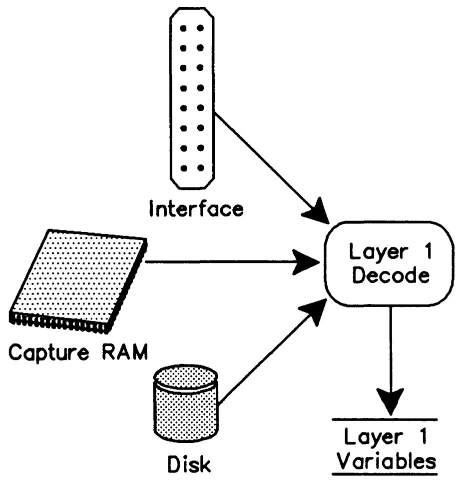


Figure 8–1 Universal Simulation/Monitor Data Flow Diagram – Decode

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

NOTE
These variables can be read with the @ (fetch) operation.

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

Example:
Determine the direction of the received stream.

```
PORT-ID @
OXFF AND        ( The AND operation eliminates the upper byte )
```

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

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

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

```
START-TIME GET_TSTAMP_MILLI
```

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

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

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

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

REC-POINTER (-- address)
Contains the pointer to the frame address field (first byte) in the received frame. Since this variable contains the address of the first byte, a double fetch operation is necessary to obtain frame contents.

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

```
REC-POINTER @  1+  C@
```

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

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

TIMER–NUMBER (-- address)
Contains the number of the expired timer. Valid values are 1 through 128.

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

Command	Error Type
OVERRUN_ERR?	Receiver overrun
CRC_ERR?	CRC error
ABORT_ERR?	Abort Error
LONG_FRM_ERR?	Frame is longer than supported by operating system buffers
SHORT_FRM_ERR?	Frame is shorter than 4 bytes including 2 CRC bytes (BOP) Improper framing (ASYNC)

Table 8–1 Error Detection

9
SIMULATION ARCHITECTURE

This section describes the structure of the Universal Simulation. The Universal Simulation program is a combination of the Universal Monitor application plus the capability of transmitting frames/blocks, lead changes, etc. via user–written test scripts.

9.1 Live Data

The simulation receives events from the interface and processes them as shown in Figure 9–1.

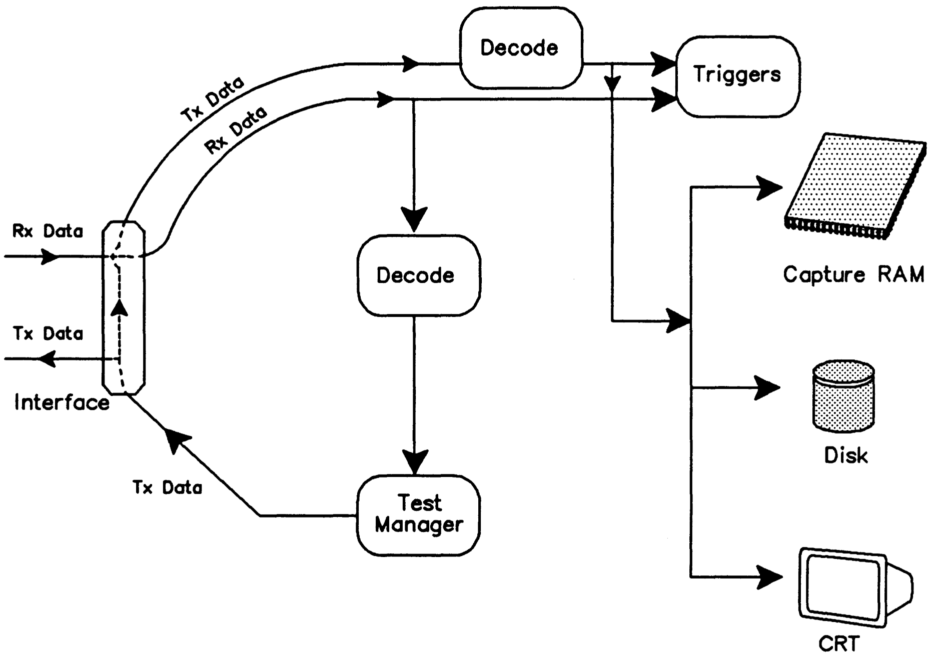


Figure 9–1 Universal Simulation Data Flow Diagram – Live Data

By default, the Universal Simulation captures the received/transmitted data in the capture RAM buffer and displays it on the screen in short format report.

Display topic
Live Data function key

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

9.2 Playback

Data can be played back from either capture RAM or disk without interfering with an active test (i.e. dropping the link) as shown in Figure 9-2.

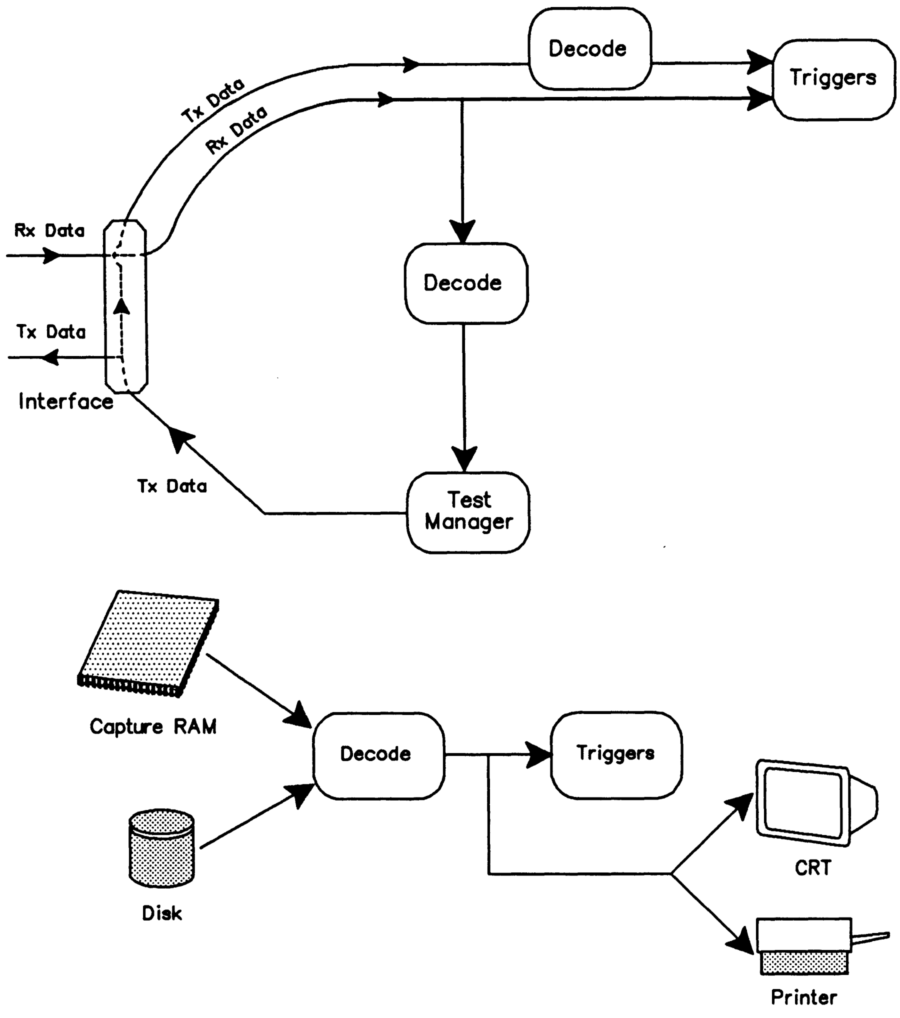


Figure 9-2 Universal Simulation Data Flow Diagram - Offline Processing

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.

9.3 Simultaneous Live Data and Playback

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

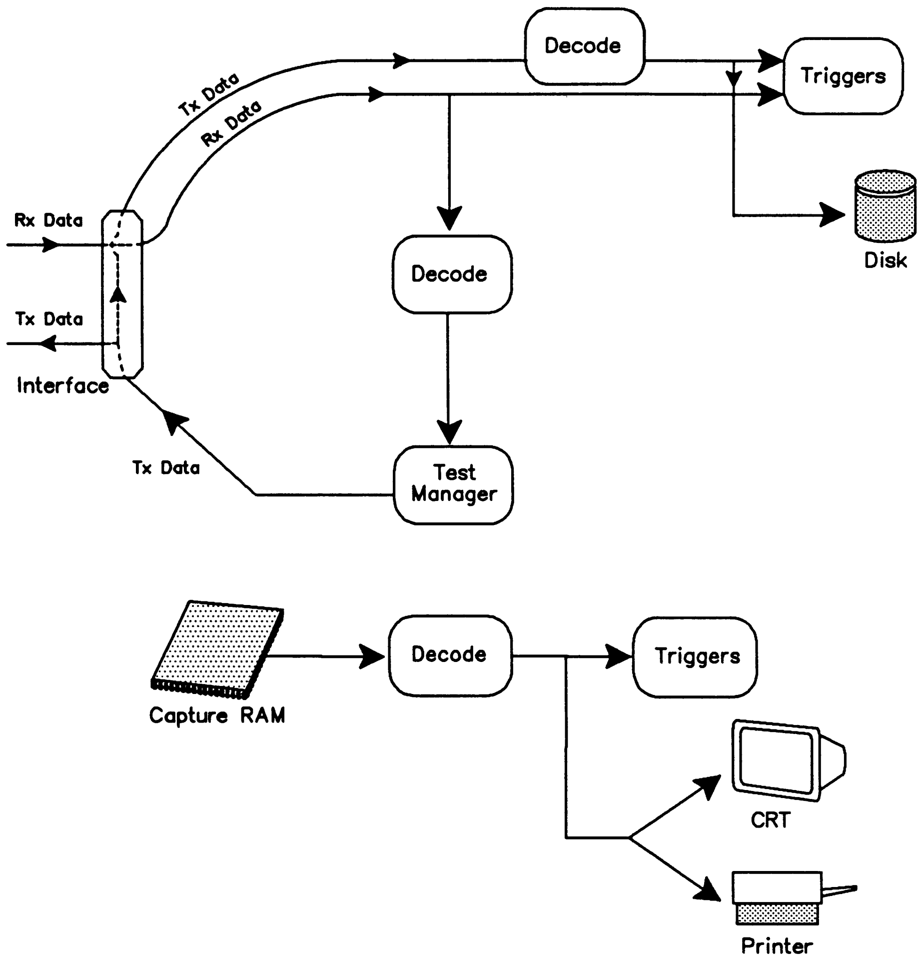


Figure 9–3 Universal Simulation Data Flow Diagram – Freeze Mode

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

FREEZE (--)
Enables data to be recorded to disk while data from capture RAM is played back.

10
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 Universal Simulation and Monitor applications.

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 event and action commands specific to the USM.

10.1 ITL Constructs

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

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

STATE_INIT{ }STATE_INIT (number --)
Brackets the execution sequence performed prior to entering a state. The initialization logic for a state is executed independently of how it was called.

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

The STATE_INIT{ }STATE_INIT clause is executed only once each time the state is entered from another state.

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

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

NEW_STATE (n --)
Executes the initialization logic of the specified state (providing STAT_INIT{ }STAT_INIT is defined) and establishes the state to be executed for the next event. Any remaining action code for the current state is then executed. It must be preceded with a valid state number and be inside the ACTION{ }ACTION brackets. This command is not mandatory if no state change is desired.

TM_STOP (--)
Stops the execution of the test script. The test suite remains in memory and can be re-executed until another test script is loaded.

SEQ{ }SEQ (number --)
Brackets a definition of tasks and outputs which execute as part of the state machine action. SEQ{ expects a single integer which is the sequence number. Up to 256 sequences are supported. Valid values are 0 through 255. The SEQ{ }SEQ partners are extremely useful when more than one action sequence calls the same tasks and outputs. The SEQ{ }SEQ definition is defined outside the ACTION{ }ACTION definition and then called by the RUN_SEQ command.

This is an alternate mechanism to generate colon definitions. This mechanism causes the equivalent of a colon definition (now accessed via a numeric identifier) to be compiled into the test script dictionary rather than the user dictionary. Refer to the Programmer's Reference Manual.

RUN_SEQ (number --)
Executes a specified set of tasks defined in a SEQ{ }SEQ definition. It is called inside an ACTION{ }ACTION definition and must be preceded with a defined sequence number.

LOAD_RETURN_STATE (number --)
Permits the test script writer to program the equivalent of subroutine calls (used with RETURN_STATE). LOAD_RETURN_STATE sets the state to which control is to be returned. LOAD_RETURN_STATE must be within the action field; nesting is not permitted.

RETURN_STATE (--)
Returns control to the state specified by LOAD_RETURN_STATE 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.

10.2 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 true, the following action clauses are evaluated in a sequential manner. Once an event evaluates true, the subsequent action clauses in that particular state are not examined.

Layer 1

If the Universal Simulation/Monitor is running on a B-Channel, no layer 1 events will be received by the test manager. See the Programmer's Reference Manual for a description of layer 1 events, i.e. control lead transitions, when the application is running on a WAN interface.

NOTE
Interface leads must be enabled.

The following commands are used to recognize data lead changes. Data lead transitions must be requested before they can be detected in a test script. The received frame indications are not affected by data lead indications.

REQ_RXD_TRANS (number --)
Requests the next specified transitions (both positive and negative) on the data lead be reported and passed to the test script. Valid values are 1 through 65535.

REQ_RXD_ON_TRANS (number --)
Requests the next specified positive transitions on the data lead be reported and passed to the test script. Valid values are 1 through 65535.

REQ_RXD_OFF_TRANS (number --)
Requests the next specified negative transitions on the data lead be reported and passed to the test script. Valid values are 1 through 65535.

NOTE
These three request transition commands are mutually exclusive. Executing one of these commands nullifies any previous request transition command.

NOTE
Receiver overflow is possible when several data lead transitions are requested and the monitor is operating at high speed.

?RXD_ON (-- flag)
Returns true if a positive transition on the data lead is received.

?RXD_OFF (-- flag)
Returns true if a negative transition on the data lead is received.

RXD-TRANS (-- address)
Contains the direction of the last data lead transition. Possible values are P_TRANS (positive transition) and N_TRANS (negative transition).

RXD_STATE (-- state)
Returns 1 if the received data lead is high, and 0 if the received data lead is low.

Received Frames

ITL provides recognition of CRC/parity errors, aborted frames, and anchored or unanchored comparison of user-defined octets.

Octets for comparison can be specified using:
- an ASCII (7 bits/no parity) string using " string";
- hex character string using X" string";
- an ASCII string sensitive to bits/character and parity. Use the MAKE_DATAn commands; or
- an EBCDIC string converted from an ASCII string using the A_TO_E command.

DATA1 (-- address)
Contains the string converted by MAKE_DATA1. This buffer contains a maximum of 255 characters.

NOTE
Similarly, the DATA2 through DATA8 buffers contain the string converted by the corresponding MAKE_DATAn command.

MAKE_DATA1 (" string"--)
Converts the specified string according to the current configuration for bits/characters and parity and stores the converted string in the DATA1 buffer. Maximum length of the string is 80 characters if entered from the keyboard and 255 characters if entered in a test script.

Example:
```
" HELLO" MAKE_DATA1
```

Send topic
String 1 function key

The following table shows the hex values for this string after conversion with different configurations.

	No Parity	Odd Parity	Even Parity
7 bits	48454C4C4F	C8454C4C4F	48C5CCCCCF
6 bits	08050C0C0F	08454C4C4F	48050C0C0F
5 bits	08050C0C0F	08252C2C2F	28050C0C0F

Table 10–1 ASCII Character Conversion

NOTE
Similarly, the MAKE_DATA2 through MAKE_DATA8 commands convert and store the string in the corresponding DATAn buffer.

A_TO_E (" string"--count\0) for successful conversion
(" string"-- -1) for failed conversion

Converts the specified string to EBCDIC and, if successful, returns the number of converted characters and 0. If unsuccessful, -1 is returned. The converted string is stored in the EBCDIC-BUF variable. The maximum string length is 80 characters if entered from the keyboard, and 255 if used in a test script.

EBCDIC-BUF (-- address)

Contains the EBCDIC string converted with the A_TO_E command. The first byte of EBCDIC-BUF is left unchanged (i.e the converted character count is not stored). .The count can be stored in the first byte after conversion.

Example:
Convert the ASCII string " HELLO" to an EBCDIC " HELLO".

```
" HELLO" A_TO_E 0=                  ( Perform conversion )
IF                                  ( Conversion was successful )
      EBCDIC-BUF C!                 ( Store the count )
ENDIF
```

Example:
Convert the ASCII string " HELLO" to an EBCDIC " HELLO" and then move the converted string to DATA1.

```
" HELLO" A_TO_E 0=                  ( Perform conversion )
IF                                  ( Conversion was successful )
      DUP                           ( Duplicate the count )
      DATA1 C!                      ( Put count in first byte of DATA1 )
      EBCDIC-BUF 1+                 ( Get converted string )
      DATA1 1+ ROT CMOVE            ( Move to DATA1 )
ENDIF
```

Send topic
String 1 function key

?RECEIVED (string -- flag)

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

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

Example:
Search for the string 'HELLO' starting at the first byte of the received frame using one of the following methods.

- ```
  " HELLO" ?RECEIVED            ( ASCII string )
  ```
- ```
  X" 48454C4C4F" ?RECEIVED      ( Hex string )
  ```
- ```
  " HELLO" MAKE_DATA1           ( Convert ASCII string )
    DATA1 ?RECEIVED             ( Use converted string )
  ```
- ```
  " HELLO" A_TO_E 0=            ( Convert string to EBCDIC )
  IF
        EBCDIC-BUF C!
  ENDIF
  EBCDIC-BUF ?RECEIVED          ( Use converted string )
  ```

NOTE
To accommodate "don't care" character positions, the question mark character for ASCII or hex 3F character can be used. The maximum string length is 80 characters. The received string can be longer than the specified string.

WARNING
These wildcard characters should not be used with the MAKE_DATAn or A_TO_E commands.

Example:
Search for the letter 'E' as the second character in a received frame or block using one of the following methods.

- ```
  " ?E" ?RECEIVED          ( ASCII string )
  ```
- ```
  X" 3F45" ?RECEIVED       ( Hex string )
  ```

?RECEIVED_DTE (string -- flag)

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

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

?RECEIVED_DCE (string -- flag)

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

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

?SEARCH (string -- flag)
Returns true if a user-defined character string is found in the received frame or block.

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

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

```
" IDACOM" ?SEARCH
```

?SEARCH_DTE (string -- flag)
Returns true if a user-defined character string is found in the frame or block received from the DTE.

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

?SEARCH_DCE (string -- flag)
Returns true if a user-defined character string is found in the frame or block received from the DCE.

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

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

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

Timeout Detection

There are 128 user programmable timers available. Timers 1 through 24 and 30 through 128 can be used in the test manager. Timer 34 is the wakeup timer. The remaining timers are used in the application and should not be started or stopped in a test script.

?TIMER (timer # -- flag)
Returns true if the specified timer has expired. Valid input parameters are timers 1 through 24 and 30 through 128.

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

```
8 STATE{
     21 ?TIMER          ( Check for timeout of timer 21 )
     ACTION{
         T." Timer 21 has expired." TCR
     }ACTION
  }STATE
```

?WAKEUP (-- flag)
Returns true if the wakeup timer has expired. The wakeup timer can be used to initiate action sequences immediately upon the test manager starting. Timer 34 is started for 100 milliseconds when the test manager is started after a WAKEUP_ON command has been issued. The default is WAKEUP_OFF.

Example:
In State 0 look for the expiration of the wakeup timer. The action is to prompt the user to press a function key, and then the test manager goes to State 1.

```
0 STATE[
      ?WAKEUP        ( Check for timeout of wakeup timer )
      ACTION[
          T." To start the test, press UF1." TCR
          1 NEW_STATE
      ]ACTION
  ]STATE
```

Function Key Detection

Refer to the Programmer's Reference Manual.

Interprocessor Mail Events

Refer to the Programmer's Reference Manual.

Wildcard Events

USM supports the OTHER_EVENT test manager command and the EVENT–TYPE variable. Refer to the Programmer's Reference Manual.

The EVENT–TYPE variable contains one of the following constants: FRAME, TIME*OUT, LEAD*CHANGE, FUNCTION*KEY or COMMAND_IND.

FRAME (-- value)
A constant value in the EVENT–TYPE variable when the received event is a frame. See the 'Received Frames' section on Page 10–4.

TIME*OUT (-- value)
A constant value in the EVENT–TYPE variable when the received frame is a timeout. The actual timer is in the TIMER–NUMBER variable. See the 'Timeout Detection' section on Page 10–7.

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

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

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

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

10.3 USM Actions

All of the general actions explained in the Programmer's Reference Manual are supported in USM.

Layer 1 Actions

The following simulation commands turn control leads on and off.

NOTE
The simulation can be configured as TO DCE or TO DTE. The commands applicable to the actual configuration are the only ones which result in a control lead transition.

V.28/RS-232C Interface		
OFF to ON	**ON to OFF**	**Description**
RTS_ON	RTS_OFF	Request to send
CTS_ON	CTS_OFF	Clear to send
DSR_ON	DSR_OFF	Data set ready
CD_ON	CD_OFF	Carrier detect
DTR_ON	DTR_OFF	Data terminal ready
SQ_ON	SQ_OFF	Signal quality
RI_ON	RI_OFF	Ring indicate
DRS_ON	DRS_OFF	Data signal rate select
TM_ON	TM_OFF	Test indicator
LL_ON	LL_OFF	Local loopback
SRTS_ON	SRTS_OFF	Secondary request to send

Table 10–2 V.28/RS–232C Interface Lead Transitions

V.35 Interface		
OFF to ON	ON to OFF	Description
RTS_ON	RTS_OFF	Request to send
CTS_ON	CTS_OFF	Clear to send
DSR_ON	DSR_OFF	Data set ready
CD_ON	CD_OFF	Carrier detect
DTR_ON	DTR_OFF	Data terminal ready
RI_ON	RI_OFF	Ring indicate

Table 10–3 V.35 Interface Lead Transitions

V.36/RS–449 Interface		
OFF to ON	ON to OFF	Description
RS_ON	RS_OFF	Request to send
CS_ON	CS_OFF	Clear to send
DM_ON	DM_OFF	Data set ready
TR_ON	TR_OFF	Data terminal ready
IC_ON	IC_OFF	Calling indicator
SR_ON	SR_OFF	Data signal rate select
RR_ON	RR_OFF	Data channel received line signal
TM_ON	TM_OFF	Test indicator
LL_ON	LL_OFF	Local loopback
SRTS_ON	SRTS_OFF	Remote loopback

Table 10–4 V.36/RS–449 Interface Lead Transitions

V.11/X.21 Interface		
OFF to ON	ON to OFF	Description
C_ON	C_OFF	Control lead
I_ON	I_OFF	Indicate lead

Table 10–5 V.11/X.21 Interface Lead Transitions

The transmit data lead can be set high or low. The line remains in the set state until the next TXD_ON, TXD_OFF, or send data command.

TXD_ON (--)
Transmits a steady space and keeps the line high.

TXD_OFF (--)
Transmits a steady mark and keeps the line low.

START–TIME (-- address)
Returns the address of the 48 bit timestamp associated with the last received data lead transition indication.

T/RXD-TIME (-- address)
Returns the address of the 48 bit timestamp when the last TXD_ON or TXD_OFF command was executed.

Transmitting Data

The following simulation commands are used to transmit frames (when the simulation is online). In HDLC/SDLC, BISYNC ASCII, or BISYNC EBCDIC framing, a CRC is calculated and appended to the transmitted frame. In ASYNC or CHARACTER SYNC framing, no CRC is calculated.

These frames can be specified using:

- an ASCII (7 bits/no parity) string using " string";
- a hex character string using X" string"; or
- a conversion of an ASCII (7 bits/no parity) string to match the current configuration for bits per character and parity using the MAKE_DATAn commands (see the 'Received Frames' section on page 10-4).

SEND (string --)
Transmits the specified string. The string is limited to 80 characters when entered from the keyboard and 255 when used in a test script.

Example:
Transmit the string " HELLO" using one of the following three methods:

```
" HELLO" SEND           ( Use ASCII string )
```
or
```
X" 48454C4C4F" SEND     ( Use hex string )
```
or
```
" HELLO" MAKE_DATA1     ( Convert ASCII string )
DATA1 SEND
```

The third method using DATA1 has the following function key equivalent.

Send topic
Send 1 function key

NOTE
For Bisync, control characters must be used for successful transmissions (refer to Figure A-3). To enter these control characters from the keyboard, precede each character by '\'.

SEND_WITH_ERROR (string --)
Transmits the specified string with a CRC error in HDLC/SDLC, Bisync ASCII, or Bisync EBCDIC framing and a parity error in async. In character sync framing, no CRC is transmitted. The maximum string length is 80 characters if entered from the keyboard, and 255 if used in a test script.

NOTE
Refer to the examples under SEND.

SEND_WITH_ABORT (string --)
Transmits the specified string with an abort status byte. The frame is truncated to a maximum of 4 characters.

NOTE
Refer to the example under SEND.

A_TO_E_SEND (string --)
Converts the specified string to EBCDIC and, if successful, transmits the converted string. If unsuccessful, the string is not transmitted and a notice is displayed. The maximum string length is 80 characters if entered from the keyboard, and 255 if used in a test script.

NOTE
Use in the same manner as SEND.

A_TO_E_SEND_WITH_ERROR (" string"--)
Converts the specified string to EBCDIC and, if successful, transmits the converted string with a CRC error (HDLC/SDLC, BISYNC ASCII, or BISYNC EBCDIC), a parity error (ASYNC), or no CRC error (CHARACTER SYNC). If unsuccessful, the string is not transmitted and a notice is displayed. The maximum string length is 80 characters if entered from the keyboard, and 255 if used in a test script.

NOTE
Refer to the example under SEND.

10.4 Using Buffers

IDACOM's test manager has 256 buffers available for creating customized frames. These buffers are numbered from 0 to 255 and can be created any size desired. However, the Universal Simulation limits the number of bytes that can be transmitted to 4170.

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

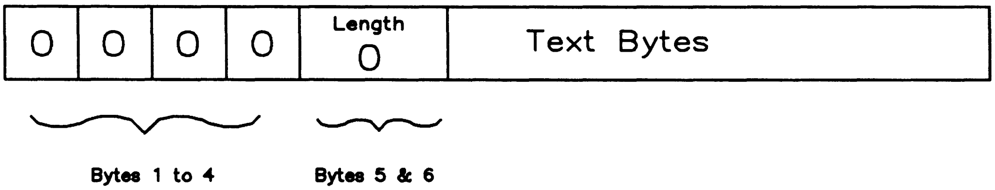


Figure 10–1 Buffer Structure

NOTE
All buffers are cleared when the TCLR command is issued. TCLR is usually the first command compiled when loading a test script.

There are three methods of moving text into a buffer.

Methods 1 and 2 automatically allocate memory for the specified text. Method 3 requires the user to allocate memory before moving text into the buffer. Use the TCLR command to clear all buffers.

Method 1

STRING->BUFFER (string\buffer number --)

Loads a quoted string into the specified buffer. The length is limited to 80 bytes if typing directly on the keyboard and 255 bytes if used within a test script. Either an ASCII or hex string can be specified. Valid buffer numbers are 0 through 255.

Example:

```
" IDACOM"  1  STRING->BUFFER          ( ASCII text moved to Buffer #1 )
X" 0100100100434445" 2 STRING->BUFFER ( Hex string of 8 bytes moved to Buffer #2 )
```

Method 2

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

Transfers a text file into the specified buffer (for text greater than 80 bytes). The file is created using the Edit function available on the Home processor. At this time, only ASCII text can be created. The last character to be transferred should be followed immediately by a CTRL 'p' character in the file. This special character is displayed as a pilcrow (¶) character. The file is transferred into the buffer until the ASCII control 'p' character is found or until the end of the file.

Example:

Create a file with the name CUSTOM.F and transfer to Buffer #3.

```
" CUSTOM.F"  3  FILE->BUFFER
```

Method 3

The following commands should not be used with FILE->BUFFER or STRING->BUFFER.

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

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

SEND_BUFFER (buffer number --)
Transmits the specified buffer. Valid buffer numbers are 0 through 255.

Example:
Create text to be included in the buffer, then transmit the buffer.
```
X" 0100100100434445" 2 STRING->BUFFER       ( Create text )
2 SEND_BUFFER                               ( Send buffer )
```

SEND_BUFFER_ERROR (buffer number --)
Transmits the specified buffer with a CRC error (HDLC/SDLC, BISYNC ASCII, or BISYNC EBCDIC), a parity error (ASYNC), and no CRC (CHARACTER SYNC).

TX–SEND–WAIT (--address)
Contains transmission queuing identifier for SEND_BUFFER and SEND_BUFFER_ERROR. When set to 0 (default), the frame is queued for transmission and the application continues. When set to 1, the application pauses until the entire buffer is transmitted.

11
TEST SCRIPTS

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

11.1 TEST1

This script is used in the simulation with either HDLC/SDLC or async framing. Set the character set to 7 or 8 bit/no parity ASCII and put the simulation online.

In state 0, on reception of a frame containing the text 'HELLO' starting at the first received character, the simulation transmits a frame containing the text 'GOODBYE' and the test manager changes to state 1.

In state 1, the reception of any frame results in the creation of a trace statement.

```
TCLR                              ( Clear test manager memory )

0 STATE{
      " HELLO" ?RECEIVED          ( Anchored match for 'HELLO' ? )
      ACTION{
          " GOODBYE" SEND         ( Transmit 'GOODBYE' )
          1 NEW_STATE             ( Go to state 1 )
      }ACTION
  }STATE

1 STATE{
      " ??" ?RECEIVED             ( Any frame received ? )
      ACTION{
          T." Frame ignored"      ( Create trace statement )
          TCR
      }ACTION
  }STATE
```

11.2 TEST2

This script is used in the simulation with either HDLC/SDLC or async framing. Set the character set to 7 or 8 bit/no parity ASCII and put the simulation online.

In state 0, on reception of a frame containing the text 'HELLO' starting at the first received character, the simulation transmits a frame containing the text 'GOODBYE' and the test manager changes to state 1.

In state 1, on reception of a frame containing the text 'GOODBYE' starting at the first received character; the state manager returns to state 0 waiting for reception of another 'HELLO'.

```
TCLR                                   ( Clear test manager memory )

0 STATE{
      " HELLO" ?RECEIVED               ( Anchored match for 'HELLO' ? )
      ACTION{
          " GOODBYE" SEND              ( Transmit 'GOODBYE' )
          1 NEW_STATE                  ( Go to state 1 )
      }ACTION
  }STATE

1 STATE{
      " GOODBYE" ?RECEIVED             ( Anchored match for 'GOODBYE' )
      ACTION{
          0 NEW_STATE                  ( Return to state 0 )
      }ACTION
  }STATE
```

11.3 TEST3

This test script behaves in a similar manner to TEST2 except that the state machine waits for three seconds before responding to a received 'HELLO'.

```
TCLR                                ( Clear test manager memory )

0 STATE{
      " HELLO" ?RECEIVED            ( Anchored match for 'HELLO' ? )
      ACTION{
          " GOODBYE" SEND           ( Transmit 'GOODBYE' )
          1 NEW_STATE               ( Go to state 1 )
      }ACTION
  }STATE

1 STATE{
      " GOODBYE" ?RECEIVED          ( Anchored match for 'GOODBYE' )
      ACTION{
          1 30 START_TIMER          ( Start timer 1 for 3 seconds )
          2 NEW_STATE               ( Go to state 2 )
      }ACTION
  }STATE

2 STATE{
      1 ?TIMER                      ( Timer 1 expired ? )
      ACTION{
          0 NEW_STATE               ( Return to state 0 )
      }ACTION
  }STATE
```

11.4 TEST4

This script is used in the simulation with either HDLC/SDLC or async framing. Set the character set to 7 or 8 bit/no parity ASCII and turn the simulation online.

Figure 11-1 shows the SDL representation of this script. In state 0, there are two valid events: an anchored match for either 'HELLO' or 'BONJOUR'.

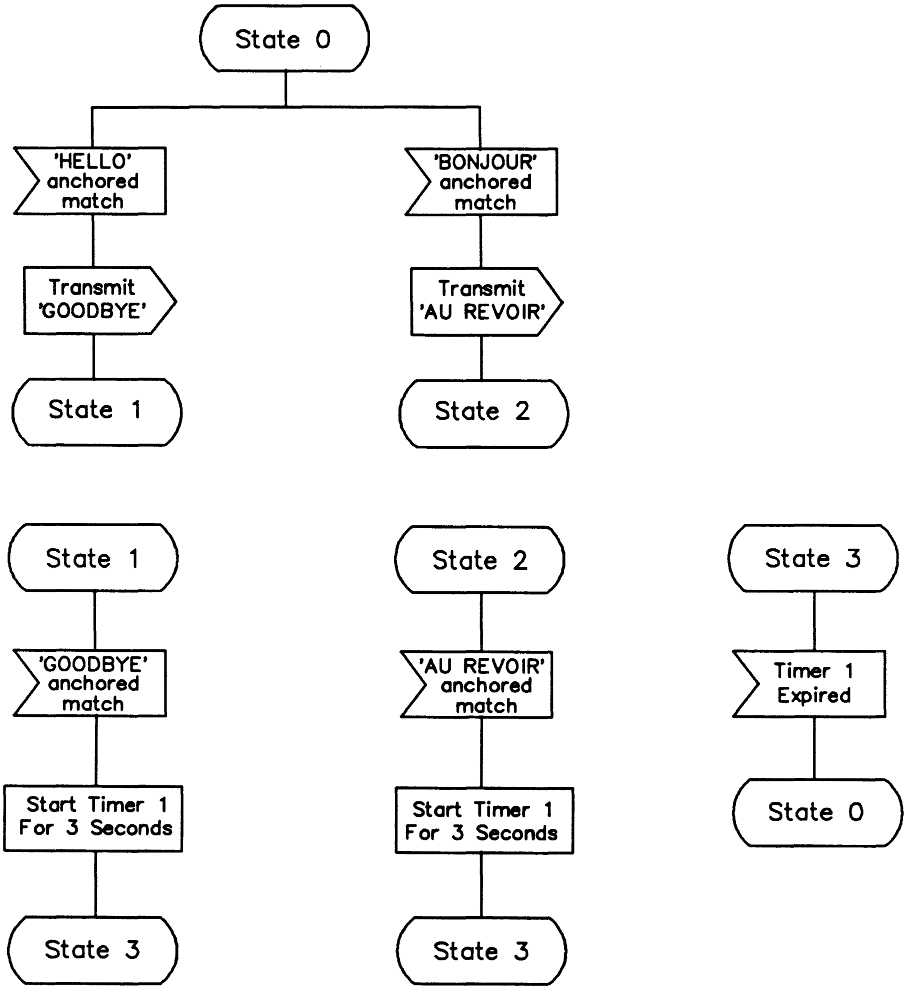


Figure 11-1 SDL Representation of TEST4

```
TCLR                              ( Clear test manager memory )

0 STATE{
      " HELLO" ?RECEIVED          ( Anchored match for 'HELLO' ? )
      ACTION{
          " GOODBYE" SEND         ( Transmit 'GOODBYE' )
          1 NEW_STATE             ( Go to state 1 )
      }ACTION

      " BONJOUR" ?RECEIVED        ( Anchored match for 'BONJOUR' ? )
      ACTION{
          " AU REVOIR" SEND       ( Transmit 'AU REVIOR' )
          2 NEW_STATE             ( Go to state 2 )
      }ACTION
  }STATE

1 STATE{
      " GOODBYE" ?RECEIVED        ( Anchored match for 'GOODBYE' ? )
      ACTION{
          1 30 START_TIMER        ( Start timer 1 for 3 seconds )
          3 NEW_STATE             ( Go to state 3 )
      }ACTION
  }STATE

2 STATE{
      " AU REVOIR" ?RECEIVED      ( Anchored match for 'AU REVOIR' ? )
      ACTION{
          1 30 START_TIMER        ( Start timer 1 for 3 seconds )
          3 NEW_STATE             ( Go to state 3 )
      }ACTION
  }STATE

3 STATE{
      1 ?TIMER                    ( Timer 1 expired ? )
      ACTION{
          0 NEW_STATE             ( Return to state 0 )
      }ACTION
  }STATE
```

11.5 TEST5

This script demonstrates the detection of control lead transitions in the simulation. Configure the simulation as TO DTE. Set the character set to 7 or 8 bits/no parity ASCII, put the simulation online.

In state 0, when the request to send lead turns off, the simulation turns the clear to send lead off, and the carrier detect lead on; starts timer 1 for one second and enters state 1.

In state 1, the test manager waits for one of two defined events. When a timeout indication is received from timer 1, the simulation transmits a frame containing the text 'HELLO WORLD' and restarts timer 1. When the request to send lead turns on, the simulation turns the clear to send lead on; the carrier detect lead off, and the test manager returns to state 0.

```
TCLR                              ( Clear test manager memory )

0 STATE{
      ?RTS_OFF                    ( Request to send lead turning off ? )
      ACTION{
          CTS_OFF                 ( Turn clear to send lead off )
          CD_ON                   ( Turn carrier detect lead on )
          1 10 START_TIMER        ( Start timer 1 for 1 second )
          1 NEW_STATE             ( Go to state 1 )
      }ACTION
  }STATE

1 STATE
      1 ?TIMER                    ( Timer 1 expired ? )
      ACTION{
          " HELLO WORLD" SEND     ( Transmit 'HELLO WORLD' )
          1 10 START_TIMER        ( Start timer 1 for 1 second )
      }ACTION

      ?RTS_ON                     ( Request to send lead turning on ? )
      ACTION{
          CTS_ON                  ( Turn clear to send lead on )
          CD_OFF                  ( Turn carrier detect lead off )
          0 NEW_STATE             ( Return to state 0 )
      }ACTION
  }STATE
```

11.6 TEST6

This script is used in the USM when configured as 'TO DCE'. Put the application online. The request to send control lead changes from OFF to ON are counted. Once forty transitions occur, a beeper is sounded and a trace statement is displayed.

```
TCLR                                    ( Clear test manager memory )

0 STATE{
      ?RTS_ON                           ( Request to send lead turning on ? )
      ACTION{
          1 COUNTER +!                  ( Increment counter )
          COUNTER @ 40 =                ( Does counter contain a value of 40 ? )
          IF                            ( Yes )
              0 COUNTER !               ( Initialize counter )
              BEEP                      ( Give audible alarm )
              T." 40 RTS leads changes"
              TCR                       ( Create trace statement )
          ENDIF
      }ACTION
  }STATE
```

11.7 TEST7

This script is used in the simulation with either HDLC/SDLC or async framing. Set the character set to 7 or 8 bit/no parity ASCII and put the simulation online.

This script shows the method of sending frames of length greater than 255 characters by using the FILE->BUFFER command.

```
TCLR                              ( Clear test manager memory )

" TEST256" 0 FILE->BUFFER         ( Transfer text into buffer 0 )

0 STATE{
      " HELLO" ?RECEIVED          ( Anchored match for 'HELLO' ? )
      ACTION{
          0 SEND_BUFFER           ( Transmit buffer 0 )
          1 NEW_STATE             ( Go to state 1 )
      }ACTION
  }STATE

1 STATE{
      " HELLO" ?RECEIVED          ( Anchored match for 'HELLO' ? )
      ACTION{
          0 NEW_STATE             ( Return to state 0 )
      }ACTION
  }STATE
```

11.8 TEST_BSC_E

This script has the same effect as TEST2. In this case, configure the simulation for Bisync EBCDIC and an unanchored match.

```
TCLR                                ( Clear test manager memory )

0 STATE{
      " HELLO" ?SEARCH              ( Unanchored match for 'HELLO' )
      ACTION{
          " SH11SXGOODBYEEX" SEND   ( Transmit 'GOODBYE' )
          1 NEW_STATE               ( Go to state 1 )
      }ACTION
  }STATE

1 STATE{
      " HELLO" ?SEARCH              ( Unanchored match for 'HELLO' ? )
      ACTION{
          0 NEW_STATE               ( Return to state 0 )
      }ACTION
  }STATE
```

11.9 PT_TEST_PAR

This test script is used with the simulation in Bisync ASCII framing. Convert the string 'HI THERE' to 7 bit ASCII/odd parity and store in DATA1. Convert the string '$^S{}_X$HI THERE$^E{}_X$' and store in DATA2.

In state 0, on reception of a frame containing the converted string 'HI THERE', the simulation transmits the converted string '$^S{}_X$HI THERE$^E{}_X$'.

```
TCLR                                  ( Clear test manager memory )

" HI THERE" MAKE_DATA1                ( Convert string to 7 bit/odd parity )
" SxHI THEREEx" MAKE_DATA2  ]         ( Convert string to 7 bit/odd parity )

0 STATE{
      DATA1 ?SEARCH                   ( Unanchored match for text in DATA1 )
      ACTION{
          DATA2 SEND                  ( Transmit text in DATA2 )
      }ACTION
  }STATE
```

11.10 PT_TEST_PAR1

This script is used with the simulation with async framing. Set the character set to ASCII and put the simulation online.

Convert the string 'HI THERE' to match the current configuration for bits per character and parity. Use the converted string both for an anchored comparison and for transmission of data.

```
TCLR                          ( Clear test manager memory )

" HI THERE" MAKE_DATA1        ( Convert string according to current configuration )

0 STATE{
     DATA1 ?RECEIVED          ( Anchored match for text in DATA1 )
     ACTION{
         DATA1 SEND           ( Transmit text in DATA1 )
     }ACTION
  }STATE
```

A
DATA FORMATS

Figures A–1 through A–5 describe the general data formats for BOP, COP, BISYNC, and ASYNC transmissions.

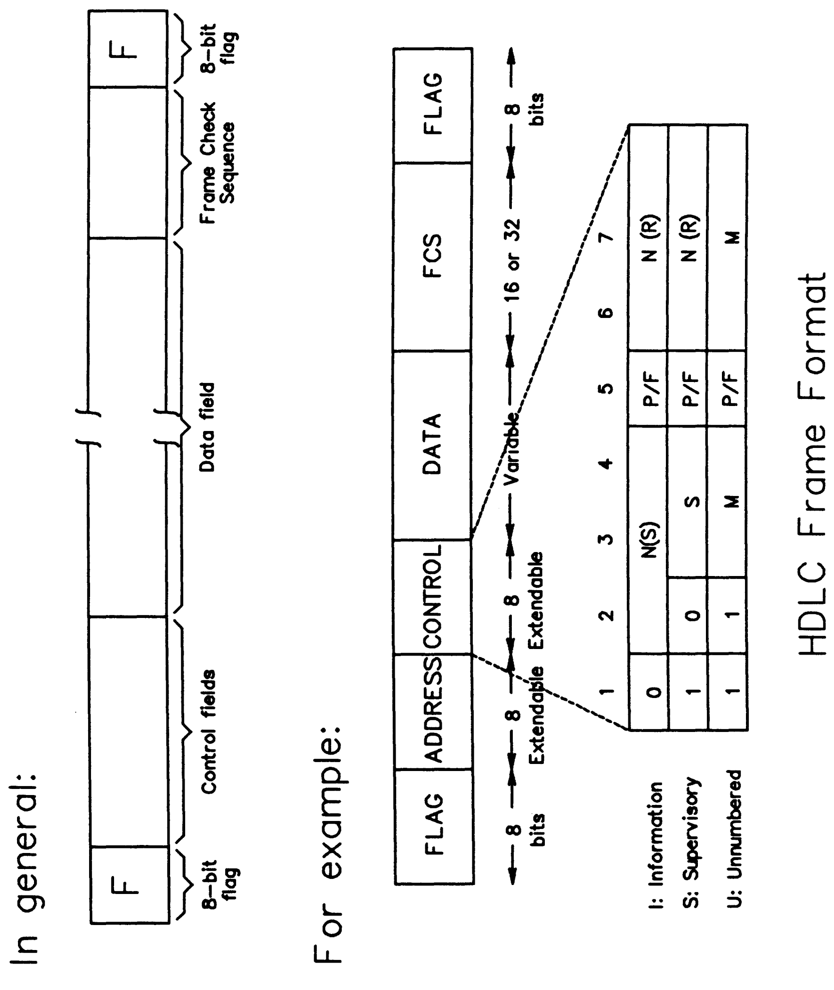


Figure A–1 Bit–Oriented Protocol Frame Format (BOP)

Control/response formats:

SYN	SYN	Control Characters

SYN	SYN	Leading Character	Control Characters

Text/header formats:

SYN	SYN	SOH	Header	ETB	BCC

SYN	SYN	SOH	Header	STX	Text	ETB/ EXT	BCC

SYN	SYN	STX	Text	ETB/ EXT	BCC

SYN	SYN	DLE	STX	Transparent Text	DLE	ETB/ EXT	BCC

Figure A–2 BISYNC Frame Formats

Mnemonic	Name	ASCII HEX	EBCDIC HEX
SYN	Synchronous Idle	16	32
SOH	Start of Heading	01	01
STX	Start of Text	02	02
ETX	End of Text	03	03
ETB	End of Transmission Block	17	26
DLE	Data Link Escape	10	10
BCC	Block Check Character		
EOT	End of Transmission	04	37
ENQ	Enquiry	05	2D

Mnemonic	Name	ASCII HEX	EBCDIC HEX
NAK	Negative Acknowledgement	15	3D
ITB	End of Intermediate Block Transmission	1F	1F
ACK 0	Acknowledgement 0	1000	1070
ACK 1	Acknowledgement 1	1001	1061
WACK	Wait for positive acknowledgement	103B	106B
RVI	Reverse Interrupt	103C	107C
TTD	Temporary Text Delay	0205	022D

Figure A–3 Control Character Descriptions

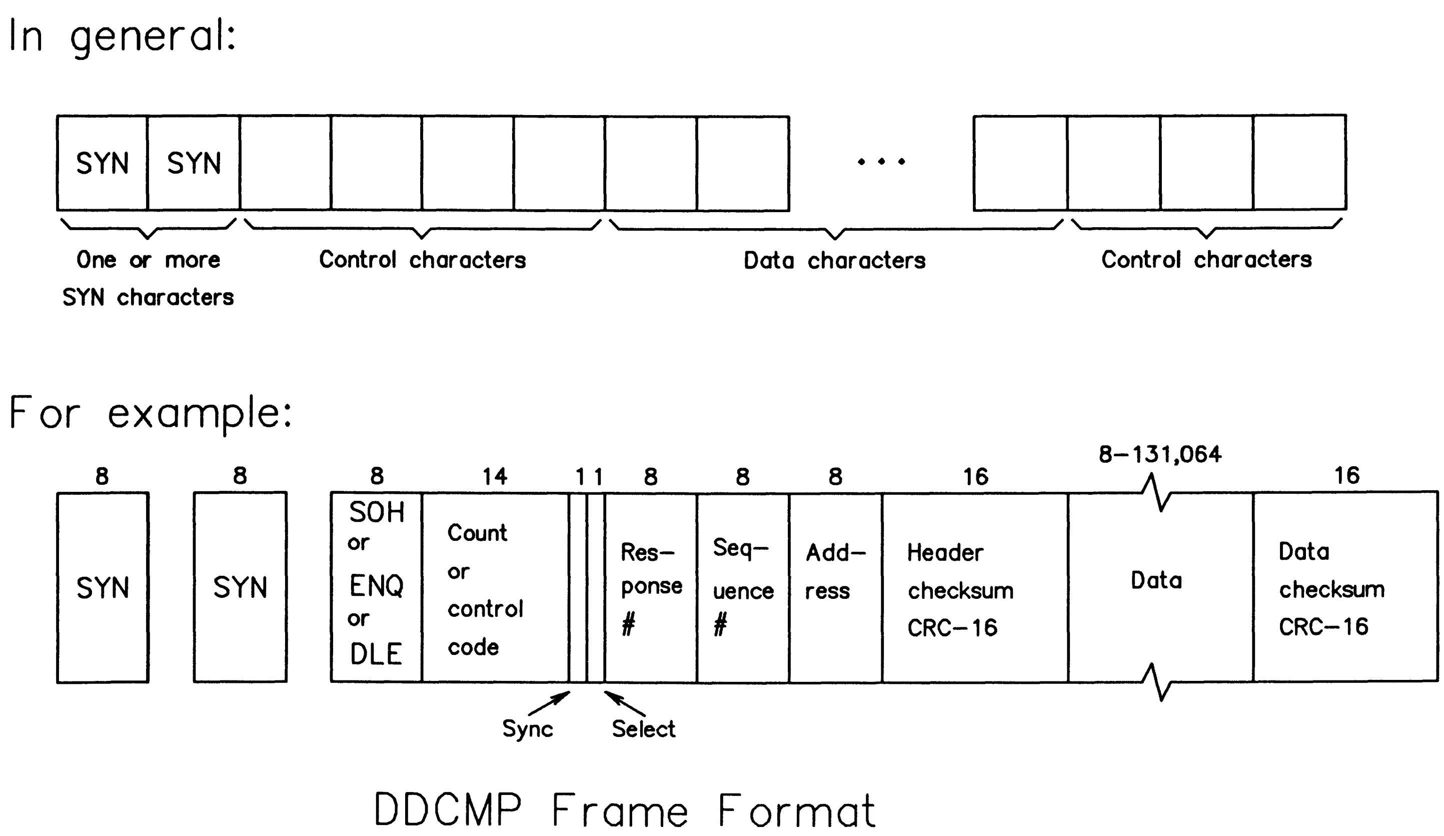


Figure A–4 Character–Oriented Protocol Transmission (COP)

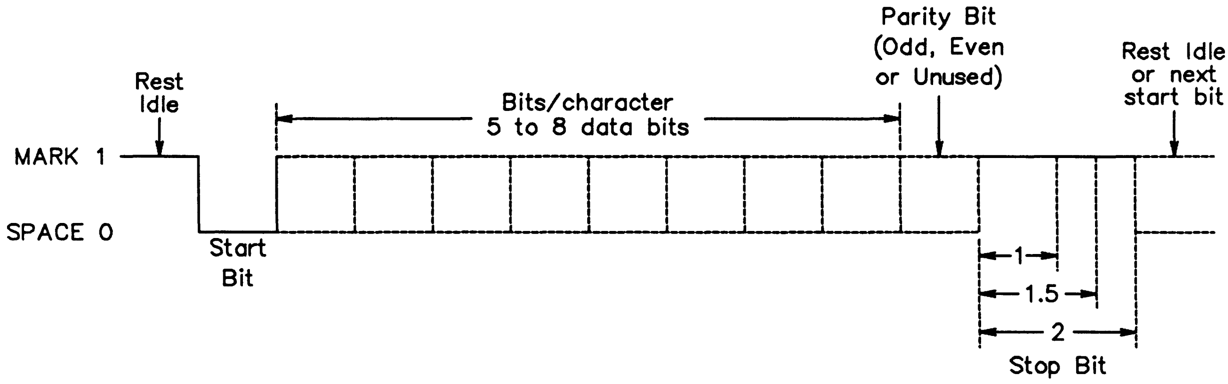


Figure A–5 ASYNC Data Character Format

The Universal Simulation and Monitor applications support two different digital signal encoding formats:

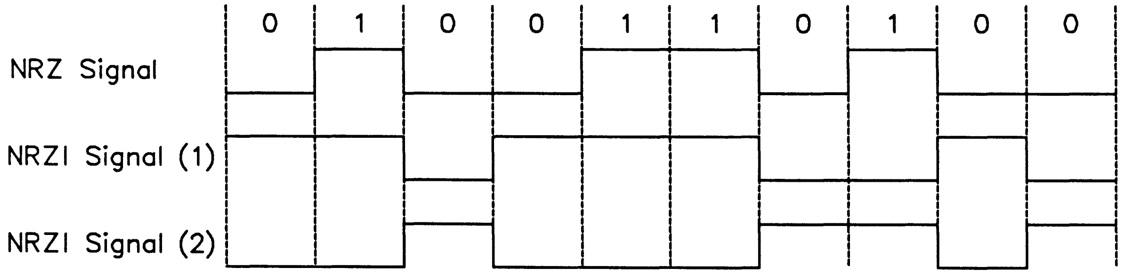


Figure A–6 NRZ and NRZI Data Encoding

Four different clocking modes are supported:

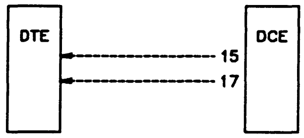
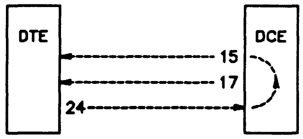
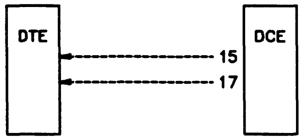
Clocking Mode	Encoding Scheme	Clocking Source
NRZ with Clock	NRZ	
External Tx Clock	NRZ	
NRZI With Clock	NRZI	
NRZI	NRZI	Clock speed is extracted from the data signal.
NRZ (Non-Return to Zero)	A 1-bit maps to a mark signal. A 0-bit maps to a space signal.	
NRZI (Non-Return to Zero Inverted)	A 1-bit maps to no transition. A 0-bit maps to a transition.	
15 – Transmit clock from DCE (DCE provided) CCITT circuit 114 17 – Receive clock from DCE (DCE provided) CCITT circuit 115 24 – Transmit clock to DCE (DTE provided) CCITT circuit 11 The pin numbers shown are for the RS-232C interface.		

Table A-1 Clocking Modes

B
COMMAND SUMMARIES

Physical Events		
Command	Stack Description	Description
?CRC_ERROR	(--)	Detects a frame with a CRC error
?ABORT	(--)	Detects an abort on the line
?RTS_ON, ?RTS_OFF	(--) (V.28, V.35)	Detects a transition on the request to send lead
?CTS_ON, ?CTS_OFF	(--) (V.28, V.35)	Detects a transition on the clear to send lead
?DSR_ON, ?DSR_ON	(--) (V.28, V.35)	Detects a transition on the data set ready lead
?CD_ON, ?CD_OFF	(--) (V.28, V.35)	Detects a transition on the carrier detect lead
?DTR_ON, ?DTR_OFF	(--) (V.28, V.35)	Detects a transition on the data terminal ready lead
?SQ_ON, ?SQ_OFF	(--) (V.28)	Detects a transition on the signal quality lead
?RI_ON, ?RI_OFF	(--) (V.28, V.35)	Detects a transition on the ring indication lead
?DRS_ON, ?DRS_OFF	(--) (V.28)	Detects a transition on the data signal rate select lead
?LL_ON, ?LL_OFF	(--) (V.28, V.36)	Detects a transition on the local loopback lead
?RS_ON , ?RS_OFF	(--) (V.36)	Detects a transition on the request to send lead
?CS_ON, ?CS_OFF	(--) (V.36)	Detects a transition on the clear to send lead
?DM_ON, ?DM_ON	(--) (V.36)	Detects a transition on the data set ready lead
?RR_ON, ?RR_OFF	(--) (V.36)	Detects a transition on the data channel signal indicator lead
?TR_ON, ?TR_OFF	(--) (V.36)	Detects a transition on the data terminal ready lead
?IC_ON, ?IC_OFF	(--) (V.36)	Detects a transition on the calling indicate lead
?SR_ON, ?SR_OFF	(--) (V.36)	Detects a transition on the data signal rate select lead
?SS_ON, ?SS_OFF	(--) (V.36)	Detects a transition on the select standby lead
?I_ON, ?I_OFF	(--) (V.11)	Detects a transition in the indicate lead
?C_ON, ?C_OFF	(--) (V.11)	Detects a transition in the control lead
?RXD_ON, ?RXD_OFF	(--) (all WAN interfaces)	Detects a transition of the data lead

Table B–1 Physical Events

Setting Leads		
Command	**Stack Description**	**Description**
RTS_ON , RTS_OFF	(--)	Sets the request to send lead (V.28, V.35)
CTS_ON, CTS_OFF	(--)	Sets the clear to send lead (V.28, V.35)
DSR_ON, DSR_ON	(--)	Sets the data set ready lead (V.28, V.35)
CD_ON, CD_OFF	(--)	Sets the carrier detect lead (V.28, V.35)
DTR_ON, DTR_OFF	(--)	Sets the data terminal ready lead (V.28, V.35)
SQ_ON, SQ_OFF	(--)	Sets the signal quality lead (V.28)
RI_ON, RI_OFF	(--)	Sets the ring indication lead (V.28, V.35)
DRS_ON, DRS_OFF	(--)	Sets the data signal rate select lead (V.28)
TM_ON, TM_OFF	(--)	Sets the test indicator lead (V.28, V.36)
LL_ON, LL_OFF	(--)	Sets the local loopback lead (V.28, V.36)
SRTS_ON, SRTS_OFF	(--)	Sets the secondary request to send lead (V.28, V.36)
RS_ON, RS_OFF	(--)	Sets the request to send lead (V.36)
CS_ON, CS_OFF	(--)	Sets the clear to send lead (V.36)
DM_ON, DM_OFF	(--)	Sets the data set ready lead (V.36)
TR_ON, TR_OFF	(--)	Sets the data terminal ready lead (V.36)
IC_ON, IC_OFF	(--)	Sets the calling indicator lead (V.36)
SR_ON, SR_OFF	(--)	Sets the data signal rate select lead (V.36)
RR_ON, RR_OFF	(--)	Sets the data channel received line signal lead (V.36)
C_ON, C_OFF	(--)	Sets the control lead (V.11)
I_ON, I_OFF	(--)	Sets the indicate lead (V.11)
RXD_ON, TXD_OFF	(--)	Sets the transmit data lead

Table B–2 Setting Leads

Frame Events		
Command	**Stack Description**	**Description**
?RECEIVED	(" string" --)	Detects a frame, anchored comparison
?RECEIVED_DTE	(" string" --)	Detects a frame from the DTE side only, anchored comparison
?RECEIVED	(" string" --)	Detects a frame from the DCE side only, anchored comparison
?SEARCH	(" string" --)	Detects a frame, unanchored comparison
?SEARCH_DTE	(" string" --)	Detects a frame from a DTE, unanchored comparison
?SEARCH_DCE	(" string" --)	Detects a frame from a DCE, unanchored comparison

Table B–3 Frame Events

Sending Frames		
Command	**Stack Description**	**Description**
SEND	(string --)	Sends a string as a frame
SEND_WITH_ERROR	(string --)	Sends a string with a CRC error
SEND_WITH_ABORT	(string --)	Sends a string and abort it during transmission
A_TO_E_SEND	(string --)	Sends a string but first convert it from ASCII to EBCDIC
A_TO_E_SEND_WITH_ERROR	(string --)	Sends a string with a CRC error but first convert it from ASCII to EBCDIC
SEND_BUFFER	(buffer number --)	Sends a buffer of data
MAKE_DATA1	(" string" --)	Converts a string from 8 bit to 5, 6, or 7 bit ASCII for transmission

Table B-4 Sending Frames

Creating Buffers		
Command	**Stack Description**	**Description**
FILE->BUFFER	(filename\buffer number --)	Loads a buffer from a file
STRING->BUFFER	(string\buffer number --)	Loads a buffer from a string (maximum 255 bytes)
ALLOT_BUFFER	(size\buffer number -- flag)	Allocates memory for a buffer
FILL_BUFFER	(data address\size\buffer number --)	Moves data into a buffer and overwrites the previous contents
APPEND_TO_BUFFER	(data address\size\buffer number --)	Appends data into a buffer
CLEAR_BUFFER	(buffer number --)	Stores a size of 0 in the buffer
BUFFER	(buffer number -- address)	Returns the address of the first byte of the specified buffer

Table B-5 Creating Buffers

Starting and Examining Timers		
Command	**Stack Description**	**Description**
START_TIMER	(timer#\time--)	Starts an alarm (countdown) timer
STOP_TIMER	(timer# --)	Stops (resets) an alarm timer
START_LAPSE_TIMER	(timer# --)	Starts an elapsed time timer
MINUTES_ELAPSED	(timer# -- minutes)	Examines the minutes elapsed for elapse time timer
SECONDS_ELAPSED	(timer# -- seconds)	Examines the seconds elapsed for elapse time timer
MILLISECONDS_ ELAPSED	(timer# -- milliseconds)	Examines the milliseconds elapsed for elapse time timer

Table B–6 Starting & Examining Timers

Timer Events		
Command	**Stack Description**	**Description**
TIMEOUT	(-- flag)	Detects a timeout of any user timer
?TIMER	(n -- flag)	Detects a timeout of a specific user timer
?WAKEUP	(-- flag)	Detects wakeup timer

Table B–7 Timer Events

Creating User Output		
Command	**Stack Description**	**Description**
T." goes to RAM and Disk too!"	(--) 1 space required after T."	Displays a timestamped comment (trace statement) in the Data Window
TCR	(--)	Inserts a carriage return with the trace statement
T.	(value --)	Displays a decimal value in the Data Window
T.H	(value --)	Displays a hexadecimal value in the Data Window
P." goes to the printer"	(--) 1 space required after the P."	Prints a comment
PCR	(--)	Sends a carriage return to the printer
P.	(value --)	Prints a decimal value
P.H	(value --)	Prints a hexadecimal value

Table B–8 Creating User Output

Program Control Events		
Command	**Stack Description**	**Description**
?KEY	(user function key # --)	Detects a function key
PROMPT" text" actions to be taken using string at address= prompt END_PROMPT	(--)	Prompts the user for keyboard input
?MAIL	(-- flag)	Detects a signal from another ITL program

Table B–9 Program Control Events

C
CODING CONVENTIONS

The following section outlines some coding and style conventions recommended by IDACOM. Although the user can develop his own style, it is suggested to stay close to these standards to enhance readability.

C.1 Stack Effect Comments

A stack effect 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:

(n_1\\n_2 -- 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:

(n_1\\n_2 -- 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.

C.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 C-1 ITL Symbols

C.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.

C.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, using 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
```

C.5 Spacing and Indentation Guidelines

The following outlines the general guidelines for spacing and indentations:

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

See the examples in Appendixes C.6 and C.4.

C.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
;
```

D
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
60	96	140	`	–
61	97	141	a	/
62	98	142	b	
63	99	143	c	
64	100	144	d	
65	101	145	e	
66	102	146	f	
67	103	147	g	
68	104	150	h	
69	105	151	i	
6A	106	152	j	|
6B	107	153	k	,
6C	108	154	l	%
6D	109	155	m	_
6E	110	156	n	>
6F	111	157	o	?
70	112	160	p	
71	113	161	q	ˆ
72	114	162	r	
73	115	163	s	
74	116	164	t	
75	117	165	u	
76	118	166	v	
77	119	167	w	
78	120	170	x	
79	121	171	y	\
7A	122	172	z	:
7B	123	173	{	#
7C	124	174	|	@
7D	125	175	}	'
7E	126	176	˜	=
7F	127	177	DEL	"
80	128	200		
81	129	201		a
82	130	202		b
83	131	203		c
84	132	204		d
85	133	205		e
86	134	206		f
87	135	207		g
88	136	210		h
89	137	211		i
8A	138	212		
8B	139	213		{
8C	140	214		≤
8D	141	215		(
8E	142	216		+
8F	143	217		†

HEX	DEC	OCT	ASCII	EBCDIC
90	144	220		
91	145	221		j
92	146	222		k
93	147	223		l
94	148	224		m
95	149	225		n
96	150	226		o
97	151	227		p
98	152	230		q
99	153	231		r
9A	154	232		
9B	155	233		}
9C	156	234		□
9D	157	235		)
9E	158	236		±
9F	159	237		■
A0	160	240		–
A1	161	241		o
A2	162	242		s
A3	163	243		t
A4	164	244		u
A5	165	245		v
A6	166	246		w
A7	167	247		x
A8	168	250		y
A9	169	251		z
AA	170	252		
AB	171	253		└
AC	172	254		┌
AD	173	255		[
AE	174	256		≥
AF	175	257		•
B0	176	260		0
B1	177	261		1
B2	178	262		2
B3	179	263		3
B4	180	264		4
B5	181	265		5
B6	182	266		6
B7	183	267		7
B8	184	270		8
B9	185	271		9
BA	186	272		
BB	187	273		┘
BC	188	274		┐
BD	189	275		]
BE	190	276		≠
BF	191	277		–

HEX	DEC	OCT	ASCII	EBCDIC
C0	192	300		{
C1	193	301		A
C2	194	302		B
C3	195	303		C
C4	196	304		D
C5	197	305		E
C6	198	306		F
C7	199	307		G
C8	200	310		H
C9	201	311		I
CA	202	312		
CB	203	313		
CC	204	314		
CD	205	315		
CE	206	316		
CF	207	317		
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		

HEX	DEC	OCT	ASCII	EBCDIC
F0	240	360		0
F1	241	361		1
F2	242	362		2
F3	243	363		3
F4	244	364		4
F5	245	365		5
F6	246	366		6
F7	247	367		7
F8	248	370		8
F9	249	371		9
FA	250	372		
FB	251	373		
FC	252	374		
FD	253	375		
FE	254	376		
FF	255	377		

E
BAUDOT CHARACTER SET

Dec Value	Binary	Hex	Unshifted Characters (letters)	Shifted Characters (figures)
0	0 0000	00	NU	NU
1	0 0001	01	E	3
2	0 0010	02	LF	LF
3	0 0011	03	A	–
4	0 0100	04	(space)	(space)
5	0 0101	05	S	'
6	0 0110	06	I	8
7	0 0111	07	U	7
8	0 0100	08	CR	CR
9	0 1001	09	D	$
10	0 1010	0A	R	4
11	0 1011	0B	J	BL
12	0 1100	0C	N	,
13	0 1101	0D	F	!
14	0 1110	0E	C	:
15	0 1111	0F	K	(
16	1 0000	10	T	5
17	1 0001	11	Z	T
18	1 0010	12	L	)
19	1 0011	13	W	2
20	1 0100	14	H	#
21	1 0101	15	Y	6
22	1 0110	16	P	0
23	1 0111	17	Q	1
24	1 1000	18	O	9
25	1 1001	19	B	?
26	1 1010	1A	G	&
27	1 1011	1B (figs)	SO (shift out)	SO (shift out)
28	1 1100	1C	M	.
29	1 1101	1D	X	/
30	1 1110	1E	V	=
31	1 1111	1F (LTRS)	SI (shift in)	SI (shift in)

Table E–1 Baudot Character Set

F
COMMAND CROSS REFERENCE LIST

This appendix cross references old commands and variables, not appearing in this manual, with new replacement commands. Reference should be made to the previous versions of this manual for description of the old commands. The new commands achieve the same function, however, the input/output parameters may have changed.

Old Command	New Command
BYTE-TIME	Contact IDACOM (Customer Support)
ONLINE	GO_ONLINE
ON_LINE	GO_OFFLINE
PLAY-COUNT	BLOCK-COUNT
PLAY-ETIME	END-TIME
PLAY-ID	PORT-ID
PLAY-STIME	START-TIME
PORT @ char SYNC_CHAR	char =SYNC char = sync character
PORT @ length EOF_COUNT	length =EOF_COUNT length = # of characters
PORT @ n SPEED	n =SPEED n = bit rate
PORT @ time ASYNC_TIME	time =ASYNC_TIME time = timeout in tenths of seconds
REC-STATUS/DATA-STATUS	STATUS_ERR?
REST=MARK	RESET_ENABLE_ON
REST=SPACE	RESET_ENABLE_OFF
SET_LONG	LONG-INTERVAL
SET_SHORT	SHORT-INTERVAL
SET_SPEED	INTERFACE-SPEED
T/RXD_TIME (received)	START-TIME

INDEX

INDEX [continued]

INDEX [continued]

INDEX [continued]