CHAMELEON 32

QUICK REFERENCE

GUIDE

Version 4.8

This version of the Chameleon 32 Quick Reference Guide
corresponds to Standard Software Release 4.3.2.

TEKELEC
26580 Agoura Road
Calabasas, California
91302

Part Number 910–3353
Part Number 909–3353

October 11, 1991

Information in this documentation is subject to change without notice. Any software which is furnished in conjunction with or embedded within the product(s) described in this documentation is furnished under a license agreement and/or a nondisclosure agreement, and may be used only as expressly permitted by the terms of such agreements(s). Unauthorized use or copying of the software or this documentation can result in civil or criminal penalties.

The hardware, software and documentation comprising the product(s) are provided under a *Tekelec limited 12 month warranty*. Other than the limited warranties that are expressly stated therein, and without limiting the generality thereof, Tekelec makes no warranty, express or implied, to you or to any other person or entity, concerning the hardware, the software and this documentation. Tekelec will not be liable for incidental, consequential, lost profits, or other similar damages, or for damages resulting from loss of use, data, revenues or time. In no event will Tekelec's liability, regardless of the form of claim, for any damages ever exceed the price/license fee paid for the specific product. You may have other rights which vary from state to state.

TABLE OF CONTENTS

CHAPTER **PAGE**

TABLE OF CONTENTS

CHAPTER **PAGE**

CONFIGURATION

This page provides brief step–by–step instructions for configuring a port for Monitoring or Simulation.

1. Power up the Chameleon. The main configuration menu should appear (see page 2).

2. Move the arrow cursor to Mode of Operation and press *F1 Monitor* or *F2 Simulat*. If you have the 256k Data Capture option, you will also have the *F4 Fast Mo* (fast monitoring) function key which enables you to monitor up to 256K bps.

3. Move the arrow cursor to the Physical Interface parameter. Press the function key that corresponds to the type of interface you are going using to use.

   a. If you selected Primary (Primary Rate Interface) or Basic (Basic Rate Interface) as your Physical Interface, press *F7 Physicl* to display the setup menu for the interface.

   b. Complete the physical interface setup menu and press *Go* to return to the main configuration menu (see pages 13 – 16 for details about these setup menus).

4. Press *F6 Setup* to display the protocol setup menu.

5. Use the function keys to select a protocol and values for the other displayed parameters. Press *Go* to return to the main configuration menu.

6. If Monitoring is your Mode of Operation, move the arrow cursor to Monitoring Data Source. Press *F1 Line* (monitor a live line) or *F2 Disk* (monitor data that has been stored to disk).

7. If Monitoring is your Mode of Operation, move the arrow cursor to Capture Mode. Press *F1 Cycle* (cyclic acquisition buffer usage) or *F2 1Buffr* (stop acquisition when buffer becomes full).

8. For Dual Port machines, follow the same steps described above to configure the second port. If you do not want to use the second port, select *Off* as its Mode of Operation.

9. Press *Go* to display the Applications Selection Menu (see page 3 for detail of menu).

10. To select an application/simulator, move the cursor arrow to the application name and press the function key that loads the application/simulator for the desired port. (The arrow cursor indicates the active window. To change between the Monitor and Simulate windows press *Shift* ↓ or *Shift* ↑.)

    If you are using the Primary Rate Interface, load the PRIMARY application to monitor the PRI during runtime. If you are using the Basic Rate Interface, load the BASIC application to monitor the BRI during runtime.

11. When you have selected all desired applications/simulators, press *Go* to start them.

    Page banners appear for each application (except Direct–to–Disk). Use the page keys shown on page 4 to display one or more pages.

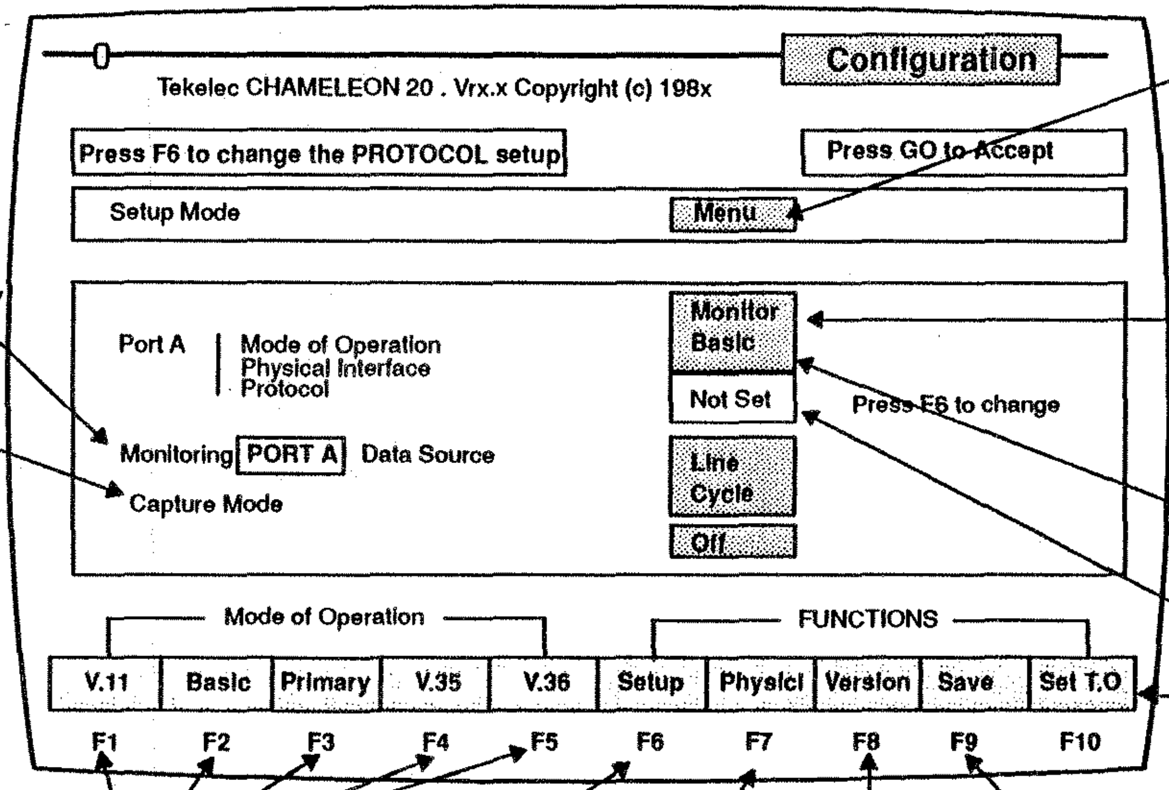
    A Simulator is indicated by a page banner named Simulate A or Simulate B. For all Simulators (except Sitrex), when you display the page, the Simulator prompt (!) appears, enabling you to enter commands and run programs immediately. To access the Simulator Parameter Set–up Menu, at the ! prompt enter the command: *setup*. Change the displayed parameters as needed, and then press *Z* to return to the ! prompt.

    In Sitrex you are taken directly to the Parameter set–up menu and not to the ! prompt. Change the displayed parameters as needed and then press *Z* to access the Simulator.

12. To stop one or more applications or Simulators, select the Configuration page, and:

    - Press *F10 Exit* to stop all applications, Simulators and the C Shell, OR
    - Move the arrow cursor to the application/simulator and press the Stop key (*F1*, *F2*, or *F3*) which stops the selected application on the desired port, OR
    - Press *F7 Reset* to restart all applications without stopping them

MAIN CONFIGURATION MENU



Configuration

Tekelec CHAMELEON 20 . Vrx.x Copyright (c) 198x

Press F6 to change the PROTOCOL setup

Press GO to Accept

Setup Mode Menu

Port A | Mode of Operation | Monitor Basic
Physical Interface
Protocol | Not Set | Press F6 to change

Monitoring PORT A Data Source | Line
Capture Mode | Cycle
Off

Mode of Operation — FUNCTIONS

V.11	Basic	Primary	V.35	V.36	Setup	Physicl	Version	Save	Set T.O
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10

Data Source:
Source of Data (Monitor mode only):
Line = Monitor a live line
Disk = Monitor data already stored to disk

Capture Mode:
Determines use of acquisition buffer:
Cycle = buffer cycles and overwrites oldest data
1Buffr = acquisition stops when buffer becomes full

Setup Mode:
Determines mode of configuration:
Menu = Manual Setup
Files = Load config.file
Autoexec = Load DEFAULT config.file

Mode of Operation:
Monitor = Monitor mode
Simulate = Simulate mode
Fast Mo = 256k
Monitoring option
Off = Port is not used

Physical Interface:
Type of interface:
V–type, PRI, BRI

Protocol:
Displays protocol menu

Set T.O:
Sets the value of the autoboot timer

F1 – F5:
Displays valid values for selected parameter

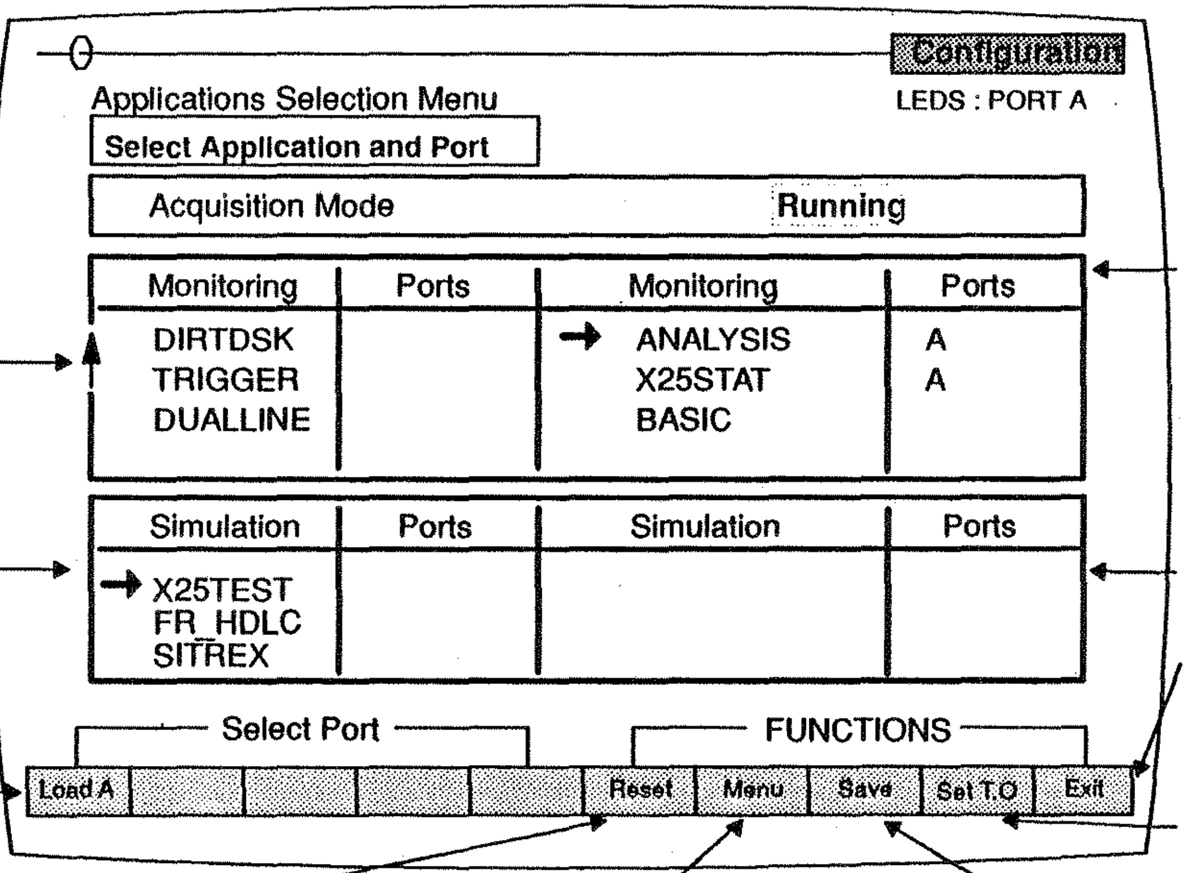
Setup:
Displays protocol menu

Physicl:
Displays setup menu for PRI (page 14) or BRI (page 12)

Version:
Displays list of system files and numbers

Save:
Saves the displayed configuration to a named file

APPLICATIONS SELECTION MENU



Applications Selection Menu

LEDS : PORT A

Select Application and Port

Acquisition Mode Running

Monitoring	Ports	Monitoring	Ports
DIRTDSK		→ ANALYSIS	A
TRIGGER		X25STAT	A
DUALLINE		BASIC	

Simulation	Ports	Simulation	Ports
→ X25TEST			
FR_HDLC			
SITREX			

Up/Down Arrow:

Indicates which arrow key to press to display additional applications in that window.

Arrow cursor:

Red arrow cursor indicates active window. White arrow cursor indicates inactive window. Press Shift ↑ or Shift ↓ to change active window

Load/Stop A:

Loads/stops the selected application.

Monitoring Window

Displayed in Monitoring and Simulation modes.

Simulation Window:

Displayed in Simulation-mode only. Displays BASIC and C simulation programs

Exit:

Stops all applications and returns to the main configuraiton menu.

Set T.O:

Sets the value of the autoboot timer.

Reset:

Resets all applications, for example; clears History buffer. Resets all statistics values to 0.

Menu:

Displays the main configuraiton menu without stopping applications.

Save:

Saves the displayed configuration to a named file.

PAGE MANIPULATION KEYS

KEY	FUNCTION
Move ↑	Moves the page banner upward one line at a time (increases the size of the page).
Move ↓	Moves the page banner downward one line at a time (decreases the size of the page).
Scroll ↑	Scrolls the data displayed in the page upward one line at a time.
Scroll ↓	Scrolls the data displayed in the page downward one line at a time.
Shift Scroll ↑	Scrolls the data displayed in the page upward the number of lines displayed in the page.
Shift Scroll ↓	Scrolls the data displayed in the page downward the number of lines displayed in the page.
Shift Hide Page	Hides the active page so that the banner is no longer visible on the screen (the application continues to run).
Show Page	Displays a page that has been hidden with Shift Hide Page.
Replace	Replaces the active page with one that has been hidden using Shift Hide Page.
Shift Move ↑	Displays the page in a special full–screen mode referred to as Blow Mode (indicated by the letter B on the top left side of the banner). Other pages cannot be accessed when the active page is in Blow Mode. Shift Move ↑ again disables Blow Mode, and returns the screen to its previous state.
Shift Move ↓	This option is available only on Chameleon 32s with PROM version 1.16 or later. When the Chameleon Remote I/O port is configured and connected to a remote device (async terminal or another Chameleon) this invokes the remote serialized mode, which transmits the active page to the remote device in serialized mode. This enables you to control the Chameleon from the remote device. See page10 for a step–by–step procedure.

FILE FORMAT AND REQUIREMENTS

Files

The Chameleon files are compatible with MS–DOS 2.x and 3.x format. File names must adhere to these MS–DOS conventions:

- File names are 1 – 8 characters in length
- Optional 1 – 3 character file extension
- Optional drive specification of A: (hard disk drive) or B: (floppy disk drive)
- File name and file extension separated by a period (.)
- Acceptable file name and path characters are:

  A–Z a–z 0–9 \ – _

Hard Disk Directories

The Chameleon includes a 40 Mbyte hard disk which provides 10 Mbytes of storage for system software and user programs, and 30 Mbytes of storage for data capture. The hard disk has the directory structure shown on the following page. The root directory can contain a maximum of 140 files. The other directories can contain a maximum of 600 files.

If the optional C Development System package is installed on your Chameleon 32, you will also have these directories: \BIN, \INCLUDE, \LIB, and \USR.

Floppy Disk Directories

Generally, when you save traffic or copy files to a floppy disk, the file is copied into the root directory of the floppy disk (unless you copy an entire *directory* to a floppy disk). When accessing a floppy disk for an application, the Chameleon searches only the root directory. Therefore all user files should be in the root directory of a floppy disk. A maximum of 112 files are permitted in the root directory of a floppy disk.

C Application Programs

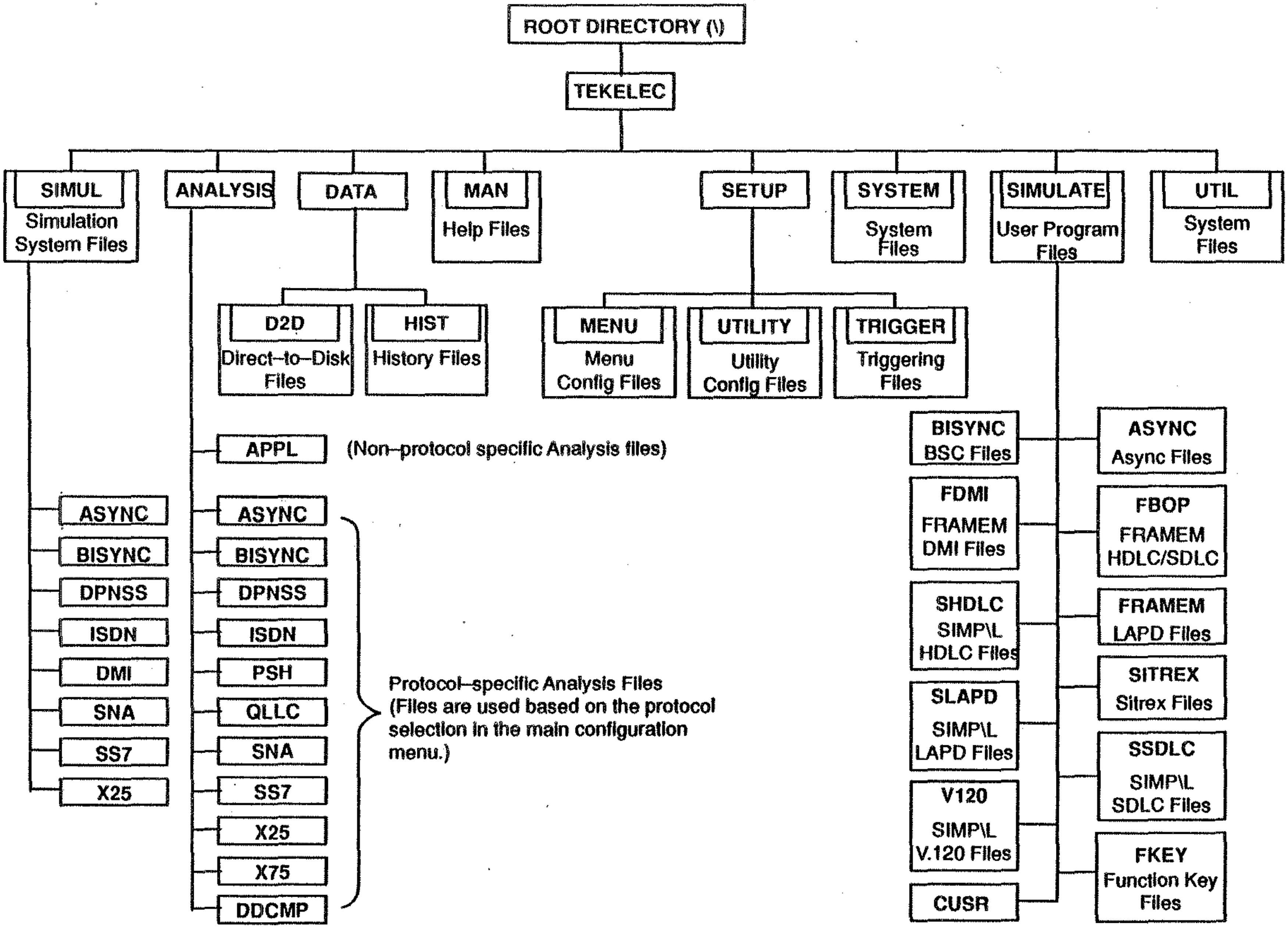
If the Chameleon 32 has the optional C package installed, you can run a C program compiled on the Chameleon 32 directly from the C shell.

You can also start a C application program from the Applications Selection menu. To do this, copy the executable file (with the file name extension .exe) to one of the following directories:

- A:\TEKELEC\ANALYSIS\xxxx (See page 6 for valid subdirectories of ANALYSIS)
- A:\TEKELEC\SIMUL\xxxx (See page 6 for valid subdirectories of SIMUL)

The directory detemines when the application will appear in the Applications Selection menu. For example, if copied to A:TEKELEC\ANALYSIS\APPL, the application will appear in the Monitoring window of the Applications Selection menu for all protocols. If copied to A:TEKELEC\ANALYSIS\X25, the application will appear in the Monitoring window of the Applications Selection menu only when X.25 is selected as the protocol. If copied to A:TEKELEC\SIMUL, the application will appear in the Simulation window of the Applications Selection menu for all protocols. The program can then be loaded and run as described on page 1.

HARD DISK DIRECTORY STRUCTURE



FILE MANAGEMENT MENU

The File Management page can be invoked at any time by pressing the *Files* key. The File Management Menu contains the following options:

F1	Chdir	Changes the current disk directory.
F2	Copy	Copies selected files to the hard disk or a floppy disk.
F3	Delete	Deletes files from the the hard disk or a floppy disk.
F4	Rename	Renames the selected files.
F5	Format	Formats floppy diskette.
F6	Disk Copy	Copies the entire contents of a floppy disk to another floppy disk.
F7	Transmit File	Transmits files to a host computer (see page 12 for more information).
F8	Receive File	Receives files from a host computer (see page 12 for more information).
F9	Connect	Establishes a connection between the Chameleon and a host computer for file transfer or host terminal emulation. See page 11 for more information.

Directory Format

There are two directory display formats. The default format displays files in four columns and shows the file name only. This format lists 60 files per screen. There is also a detailed directory format, which displays the time, date, and size of the file. The detailed format displays 15 files per screen in a single column.

To toggle between the two display formats, press *Ctrl D.*

If the directory is longer than one screen display, the page number appears in the upper right corner. To move to the previous or next screen, press *Shift* ↑ or *Shift* ↓, respectively.

List Selector

The List Selector enables you to select several files or sub–directories for a single file management operation. To use the List Selector, do the following:

1. When the disk directory is displayed, move the arrow cursor to the first file or directory you want to select.
2. Press the space bar to highlight the file or directory name. (To unselect a file, press the space bar a second time.)
3. Continue this procedure to highlight all desired files.
4. Select the file management operation. For example, press *F2 Copy* to copy the selected files.

View ASCII File(s)

After opening a directory to the file level, you can view the text of one or more of the files. This is possible only for ASCII files, not for directories or binary files.

To view ASCII files:

1. Move the cursor to the desired file.
2. Press the spacebar to highlight it.
3. Repeat steps 1 and 2 for any additional files.
4. Press *Ctrl V.* The selected file(s) are opened. F–keys 1 through 5 take on special functions:

F1 MORE	Scrolls down 1 page of current file.
F2 NEXT	Returns to files list or to start of next file
F3 PREV	Jumps to start of previous file, or current one if only one file open.
F4 RESTART	Jumps to start of current file.
F5 QUIT	Quits to directory.

UTILITIES MENU

The Utilities page can be invoked an any time by pressing *Shift Utilities*. The File Utilities Menu contains the following options:

F1 Remote I/O Port Setup Configures the Remote I/O port so that an Async terminal can be used to control the Chameleon remotely.

F2 Printer Setup Configures a printer port to output to a serial or parallel printer. See page 9 for a list of print commands and keys.

F3 Set Date and Time Sets the system time and date.

F4 Traffic Load/Save Saves Direct–To–Disk or Acquisition buffer traffic to a file. Loads a traffic file for Monitoring.

F5 645/705 Data Conversion Converts data acquired over a V–type interface by HARD Engineering Models 645 and 705 to a format compatible with the Chameleon 32..

F6 Check Free Disk Space Displays the number of bytes available on the hard disk or a floppy disk.

F7 Kermit/Connect Mode Setup Configures the Aux Serial Port 2 for Kermit File Transfer. See pages 12 for more information.

F8 Backup/Restore Menu Backs up the entire hard disk or files that are larger than one floppy (700 Kbytes).

F9 FMS File Conversion Converts files created with the Chameleon 32 FMS operating system (software release 2.6.1 or earlier) to the Chameleon MS–DOS operating system format (Release 3.0 and later).

PRINT KEYS AND COMMANDS

APPLICATION	KEY/COMMAND	RESULT
All applications	Print Scrn key Print Page key	Prints the current screen Prints the active page
History	Ctrl P	Displays print menu to print a user–defined range of events. See page 16 for a complete description.
X.25 Statistics	F3 Print key	Prints an X.25 statistical report
SNA Statistics	F3 Print key	Prints an SNA statistical report
BSC Statistics	F2 Print key	Prints a BSC statistical report
ISDN Statistics	F5 Print key	Prints an ISDN statistical report
SS#7 Statistics	F1 Print key	Prints an ISDN statistical report
BASIC Simulators	LFILES command LFLIST command LLIST command LMLIST command LPRINT command LTPRINT command	Prints file directory Prints current function key assignments Prints the program in memory Prints the mnemonic table in memory Prints text Prints the contents of the trace buffer
SITREX	LDISPT command LDISPC command LDISPV command LDISPX command LDISPM command LLIST command LPRINT command	Prints timer values in decimal Prints counters in hex Prints variable values Prints numeric variables in hex Prints length and contents of message buffer Prints the scenario in memory Prints text
C Shell	>.PRT	Redirects output to the printer
	Aux Serial Port 2 Library Functions	See page 58 for a description of the functions.
Triggering	ACTION= STATS PRINT	Prints the Statistics report when the triggering condition is met

REMOTELY CONTROLLING THE CHAMELEON

Setting up your Chameleon to be controlled from a remote device entails two basic procedures: configuring the Chameleon as the slave (remotely–controlled) device, and configuring another device as the master. This master controller may be another Chameleon or other terminal device, such as a PC. The remote mode supports the Chameleon multiple page capability. **To maximize the performance of the Chameleon, always disconnect a remote terminal when not in use.**

To set up your Chameleon as a slave device:

1. Using an RS–232 cable, connect the Chameleon to the master device:
    - If a terminal (async or PC terminal emulation) is the master device, connect the cable to the Chameleon Remote I/O port
    - If another Chameleon is the master device, connect a null–modem cable to the Aux 2 port on the master Chameleon, and to the Remote I/O port on the slave Chameleon
2. At the slave Chameleon, open the Utilities menu and select ***F1 Remote I/O Port Setup.***
3. Configure the slave Chameleon to transmit by selecting the parameter values (terminal type, baud rate, data bits, etc) required by the remote device.
4. Press *Go* to accept the parameter values and to start remote mode. The Chameleon can then be accessed using the keys shown in the table on the next page.

To set up your Chameleon as a master device:

1. Open the Utilities menu and configure the KERMIT/Connect mode (F7) to match the slave (remote) device.
2. Press *Go* and exit from the Utilities menu.
3. Open the File Management menu and press ***F9–Connect.***
4. Press ***TAB, TAB*** to re–fresh the screen and display data as displayed by the slave device.

To disable the remote control of your master, press ***Shift Cancel.***

Once in remote mode, an alternate, serialized, remote mode can be activated. This causes the remote terminal screen to be updated constantly. However, only the active page is displayed by the remote terminal.

To activate/deactivate serialized remote mode:

1. At the master device, press ***Shift Move***
2. At the slave device, press ***Tab Shift F***

The letter **R** in the banner of the active page on the slave device indicates that you are functioning in the serialized remote mode.

To emulate the Chameleon key:	On the host, use:	Hex Code	To emulate the Chameleon key	On the host, use:	Hex Code
F1	Tab 1	09 81	Scroll ↑	Tab g	09 67
F2	Tab 2	09 82	Move ↓	Tab f	09 66
F3	Tab 3	09 83	Scroll ↓	Tab h	09 68
F4	Tab 4	09 84	Left Arrow	Ctrl H	08
F5	Tab 5	09 85	Down Arrow	Ctrl J	0A
F6	Tab 6	09 86	Right Arrow	Ctrl L	0C
F7	Tab 7	09 87	Up Arrow	Ctrl K	0B
F8	Tab 8	09 88	Replace	Tab D	09 44
F9	Tab 9	09 89	Select	Tab d	09 64
F10	Tab Ctrl J	09 8A	Files	Tab b	09 42
Cancel	Ctrl X	18	Utilities	Tab B	09 62
Go	Ctrl Y	19	Run/Stop	Tab 0	09 80
Move ↑	Tab e	09 65	Space bar	Space bar	20
Print Page	Tab A	09 41	ESCape	ESCape	1B
Print Scrn	Tab a	09 61	Return	Return	0D
Hide Page	Tab C	09 43	Help	Ctrl W	17
Show Page	Tab c	09 63	Delete	Delete	7F
Shift ↑	Tab Ctrl L	09 0C	Shift ↓	Tab Ctrl N	09 0E

* If no page banner is displayed, the subject page cannot be printed out.

Chameleon Keyboard Hex Values

TERMINAL EMULATION

This procedure describes how to use the Chameleon to emulate a host terminal.

1. Connect the host to the Chameleon Aux Serial Port 2 using an RS232 cable. (The Chameleon will act as the DCE. For this reason, you may require a special RS232 cable configuration. Refer to page 112 for details.)
2. Use the Kermit/Connect Mode Setup in the Utilities menu to configure the Chameleon to be compatible with the host.
3. When the configuration parameters are set, press *Go* to accept the values.
4. On the Chameleon, invoke the File Management menu and make it active.
5. Press *F9 Connect*. This causes the Chameleon screen to go blank and behave as a host terminal.

   You can now enter host commands. To transfer files between the Chameleon and the host, refer to page 12.

6. To exit the Connect window, press *Shift Cancel*.

KERMIT FILE TRANSFER

To use the Kermit file transfer facility:

1. Verify that the host has a file transfer utility that is compatible with the KERMIT protocol.

2. Connect the host to the Chameleon Aux 2 port using an RS232 cable. (The Chameleon will act as the DCE. For this reason, you may require a special RS232 cable configuration. See page 82.)

3. Using the Kermit/Connect Mode Setup in the Utilities menu, configure the Chameleon for file transfer.

Note: Kermit automatically uses 8 data bits, 1 Stop bit and no parity, regardless of how you configure them in the Kermit/Connect Mode Setup menu. If you configure the Chameleon for terminal emulation, disregard these parameters in the Kermit/Connect Mode Setup menu. *However, you must select the type of file you are going to transfer: Text or Binary.* You cannot transmit binary and text files at the same time.

4. Call up the host Kermit program. A prompt indicates that the file transfer program has been loaded and KERMIT commands that will be executed. (When entering host commands, you can enter the commands on a host terminal *OR* you can use the Chameleon Connect window to emulate a host terminal. See page 11.)

5. On the Chameleon, open and activate the File Management menu.

6. Follow the appropriate instructions in the table below depending on whether you are transmitting or receiving files. As the file is transferred, information is displayed in the Transmit/Receive page so that you can monitor the progress of the transmission. When the transfer is complete, the screen displays the message Reception OK.
   - If the transfer fails, retransmit the file(s) by pressing *F1 Retry.*
   - If an error was detected during the file transfer, the following message appears Send failed.

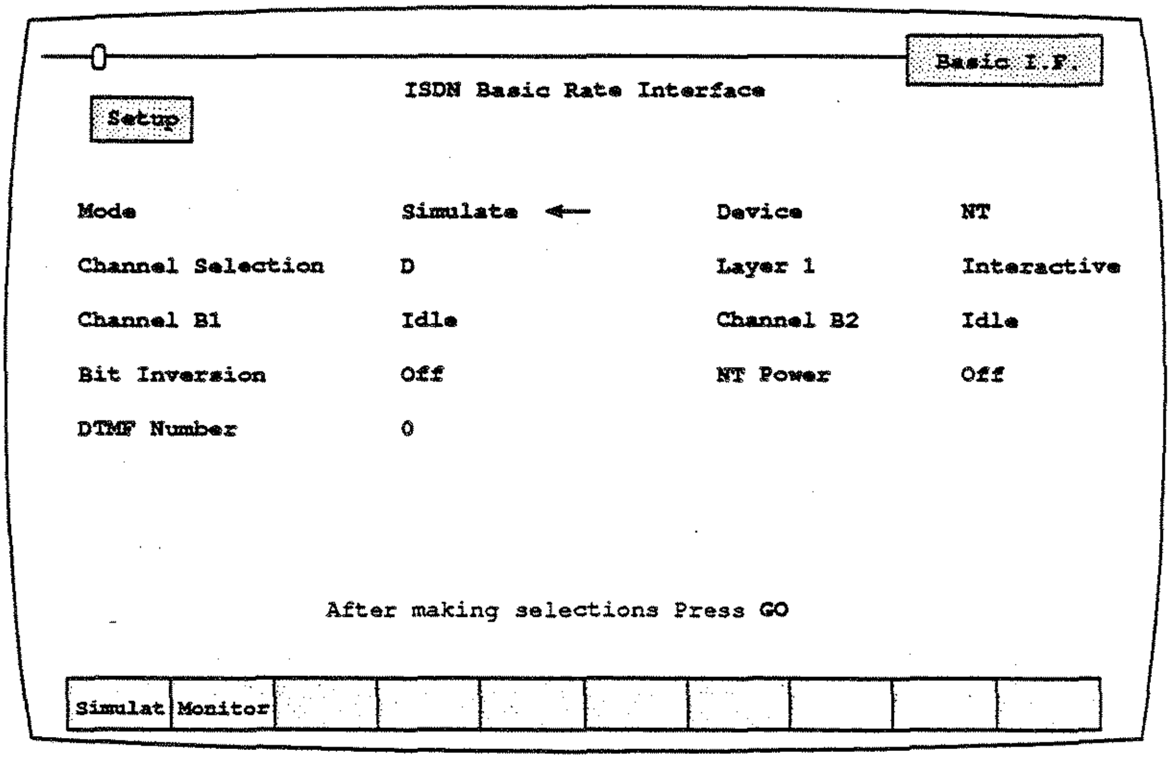
IF THE CHAMELEON IS TRANSMITTING FILES:	IF THE CHAMELEON IS RECEIVING FILES:
a. If necessary, use *F1 Chdir* in the Chameleon File Management menu to select the drive and directory that contains the files you want to transmit to the host	a.. Enter the host command that transmits the files. For example: *send filename.ext* <RETURN> [You can use the asterisk (*) as a wildcard to select more than one file to transmit. For example, to transmit all files from the host with the extension .doc, enter: *send *.doc*]
b. Use the List Selector to select the files you want to transmit. (To use the List Selector, use the arrows keys to move the red arrow cursor to the desired file, and then press the space bar to highlight the file in red. Press the space bar a second time to unselect the file.)	b. Make the File Management page active.
c. Enter the following command on the host computer: receive <RETURN>	c. Press *F8 RX File.* This begins reception.
d. In the Chameleon 32 File Management menu, press *F7 TX File.* This begins the transmission.	

7. When file transfer is complete, press *F10 Exit* to return to the File Management menu.

To abort the operation in the middle of a transfer:

Press *Esc.* The message Send failed is displayed.

Basic Rate Interface Setup Menu



ISDN Basic Rate Interface

Mode	Simulate ←	Device	NT
Channel Selection	D	Layer 1	Interactive
Channel B1	Idle	Channel B2	Idle
Bit Inversion	Off	NT Power	Off
DTMF Number	0		

After making selections Press GO

Mode Selects the mode to use for layer 1.

Simulate Simulates layer 1 as an NT or as a TE. (default)

Monitor Monitors layer 1.

Channel Selection Selects the channel to be used for running the upper level (above layer 1) protocol software. The options are: *B1, B2, or D (default).*

Device Selects the type of device to be simulated (Simulation mode only).

NT Network Termination (default)

TE Terminal Endpoint

Channel B2
Channel B1 Selects an option for the B1 or B2 channel.

Milliwatt Inserts a digital milliwatt tone (0dBm,1004 Hz ?–law or 1020 Hz A–law) in the selected B–channel. (Simulation mode only.)

Codec Allows a handset using Codec to be connected to the Chameleon Basic Rate Interface.

Idle Idles the channel with all ones data. (default)

Ext B Selected B channel available at the Ext B interface (RS422 compatible) of the Chameleon Basic Rate Interface.

Basic Rate Interface Setup Menu (continued)

Layer 1 Selects an option for layer 1 activation.

Interactive At runtime, interactive transmission of signals is possible. (No automatic activation is done.) (default)

Automatic Whenever Layer 1 is deactivated, or goes to error state, the system automatically activates.

Note The following three parameters are supported only on machines with Basic Rate Interface Board (805–0259), Revision F.

Bit Inversion Inverts the data bits when a B channel is selected for the Channel Selection parameter.

NT Power Specifies the type of power provided from the NT to the TE.

SRC1Nor Power source 1 under normal conditions.

SRC1Rev Power source 1 under emergency conditions (reverses polarity).

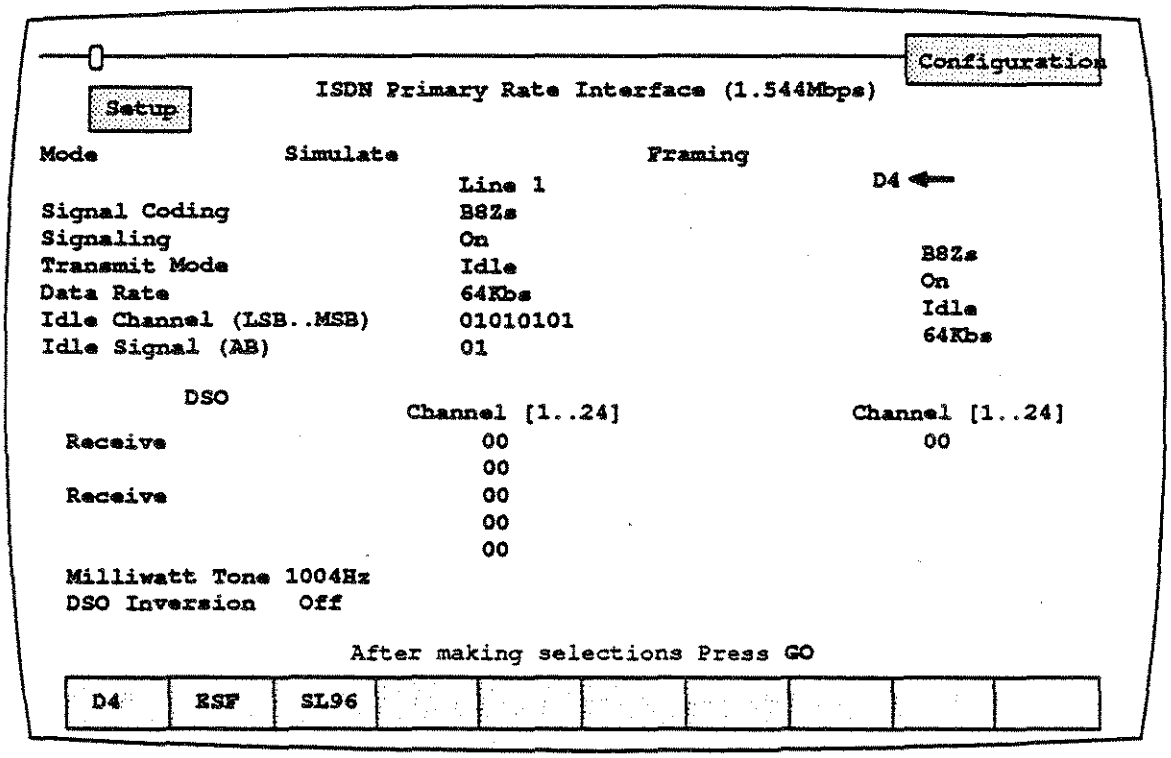
SRC2Nor Power source 2 under normal conditions.

SRC2Rev Power source 2.under emergency conditions (reverses polarity).

Off The NT power lines are turned off.

DTMF Number This parameter is relevant when the Codec unit is selected for a B–channel. It causes the Chameleon to generate the Dual Tone Multi–Frequency (DTMF) tones corresponding to the numbers entered in this field. You can enter a maximum of 20 digits in the DTMF Number field. Only digits are allowed.

Primary Rate Interface Setup Menu



```
                                                                  Configuration
 Setup          ISDN Primary Rate Interface (1.544Mbps)

Mode                 Simulate                      Framing
                                     Line 1                         D4 ◄─
Signal Coding                        B8Zs
Signaling                            On                               B8Zs
Transmit Mode                        Idle                             On
Data Rate                            64Kbs                            Idle
Idle Channel (LSB..MSB)              01010101                         64Kbs
Idle Signal (AB)                     01

         DSO
                                 Channel [1..24]                Channel [1..24]
  Receive                            00                               00
                                     00
  Receive                            00
                                     00
                                     00
  Milliwatt Tone 1004Hz
  DSO Inversion   Off

                       After making selections Press GO

 | D4 | ESF | SL96 |    |    |    |    |    |    |    |
```

Mode Selects the mode to use.

Simulate Generates data from the Chameleon, and sends it on the line. If you selected Monitor for Mode of Operation, the Chameleon simulates the physical layer, while monitoring layers 2 and above.

Monitor The Chameleon monitors the line only.

Framing Selects the type of framing to be used.

D4 D4 Framing. This is available only for the ANSI PRI.

ESF Extended Super–Frame. Available only for ANSI PRI.

SL96 Selects SLC–96 framing. Available only for ANSI PRI.

CEPT CEPT recommended framing. Available only for CEPT PRI.

Signal
Coding Selects the Zero Suppression scheme.

ANSI options: *B8ZS* Bipolar 8 Zero Suppression

*AMI*Alternate Mark Inversion, without zero suppression

CEPT options: *HDB3* High Density Bipolar 3 with zero suppression

*AMI*Alternate Mark Inversion, without zero suppression

Signaling For ANSI, enables/disables signaling information for Lines 1 and 2. For CEPT, enables signaling information in time slot 16. When Signaling is On, the Idle Signal parameter determines the idle pattern to be used.

Primary Rate Interface Setup Menu (continued)

Transmit Mode

Resync Re–synchronizes the line.

Idle Transmits an Idle Sequence on the line. Specify the idle sequence using the Idle Channel parameter. If BERT error insertion rate is set to 1.0E–3 or greater, you will encounter a frequent loss of synch and the BERT error statistic will be thrown off by the on/off loss of synch.

Transparency Available for Line 1 only. The Chameleon synchronizes the Tx clock to the Rx clock and transmits its own data *unless* the application is configured to transmit the received data.

Remote Alarm Transmits the Remote Alarm signal. (CEPT only)

Yellow Alarm Transmits the Yellow Alarm signal. (ANSI only)

Repeater The Tx clock is synchronized to the Rx clock and received data is re–transmitted.

Data Rate Sets Data X and Data Y for either 56K or 64K (ANSI only).

CRC Enables/disables a Cyclic Redundancy Check in the signals (CEPT only).

Idle Channel Specifies the idle sequence to send on channels for which no other function is selected. Enter an 8–bit sequence (LSB —> MSB).

Idle Signal When Signaling is enabled (ON), this specifies the sequence of bits to send on the signaling channel, when idle. For D4 and ESF framing, enter a two– bit pattern. For ESF, the two bits are repeated in the four bit signal. For CEPT framing, enter a four bit pattern.

The remaining parameters allow you to make selections for a specific channel/time slot. A value of 00 de–selects the current selection. For ANSI, these parameters accept a value for the channel number (1–24). For CEPT, these parameters accept a value for the time slot (1 – 31).

Receive Data X Selects the receive channel/time slot for Simulation or Monitoring packages. Data can be received on either Line 1 or Line 2, but not on both simultaneously.

Receive Codec Selects the channel/time slot to enable the Codec Receiver (Line 1 only).

Receive Data Y Selects the channel/time slot to enable the Data Y Receiver (Monitor Mode only, Line 1 only).

Transmit Data Y Simulate Mode of Operation only. Selects the channel/time slot to be used with the DS0 Y Receiver in Monitor Mode. In Simulation Mode, this parameter takes the channel/time slot for the Line 1 Transmitter.

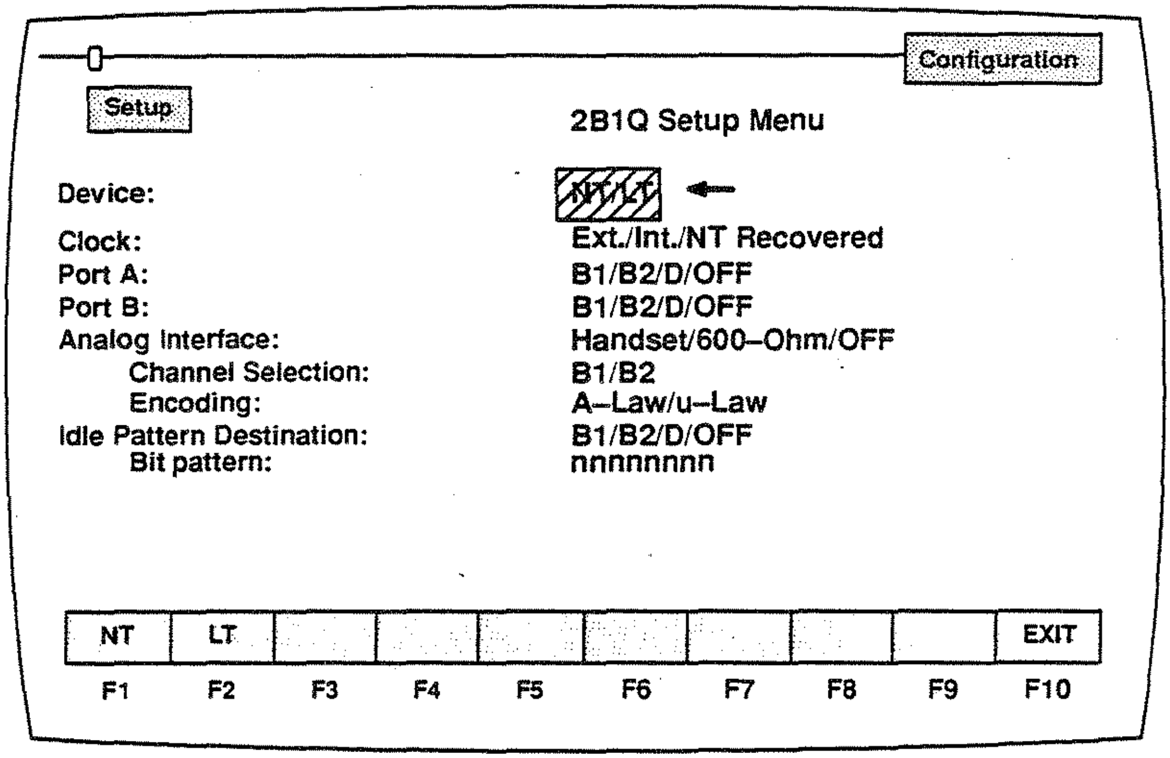
Transmit Codec Simulate mode only. Selects the channel/time slot for the Codec Transmitter.

Transmit Milliwatt Simulate mode only. Selects the channel/time slot to enable the Digital Milliwatt Tone Generator. The tone generated in ANSI (D4/ESF) is 1004Hz at 0 dBm. In CEPT, the tone can be either 820Hz or 1020Hz.

Milliwatt Tone Simulate mode only. Selects the Milliwatt Tone for CEPT.

DS0 Inversion Inverts the data on the specified channels/time slots in Data X and Data Y.

2B1Q U–INTERFACE SETUP MENU



2B1Q Setup Menu

```
Device:                          NT/LT
Clock:                           Ext./Int./NT Recovered
Port A:                          B1/B2/D/OFF
Port B:                          B1/B2/D/OFF
Analog Interface:                Handset/600–Ohm/OFF
      Channel Selection:         B1/B2
      Encoding:                  A–Law/u–Law
Idle Pattern Destination:        B1/B2/D/OFF
      Bit pattern:               nnnnnnnn
```

Device Sets your Chameleon to emulate a network (LT) or network node/terminal device (NT). This setting cannot be changed in the Run Time configuration menu.

Clock Sets your Chameleon to take its timing from an external timing source, the Chameleon 8–MHz clock (internal), or to derive clocking from the bit–stream being sent over the U–interface. This setting cannot be changed in the Run Time configuration menu.

Port A Sets Port A of your Chameleon to function as either a B1–, B2–, or D–channel port. Also deactivates Port A altogether (OFF). The channel you assign here canot also be assigned to Port B and/or the Analog Interface.

Port B Same as for Port A. The channel you assign here cannot also be assigned to Port A and/or the Analog Interface.

Analog Interface Sets the physical/electrical mode of the analog interface.

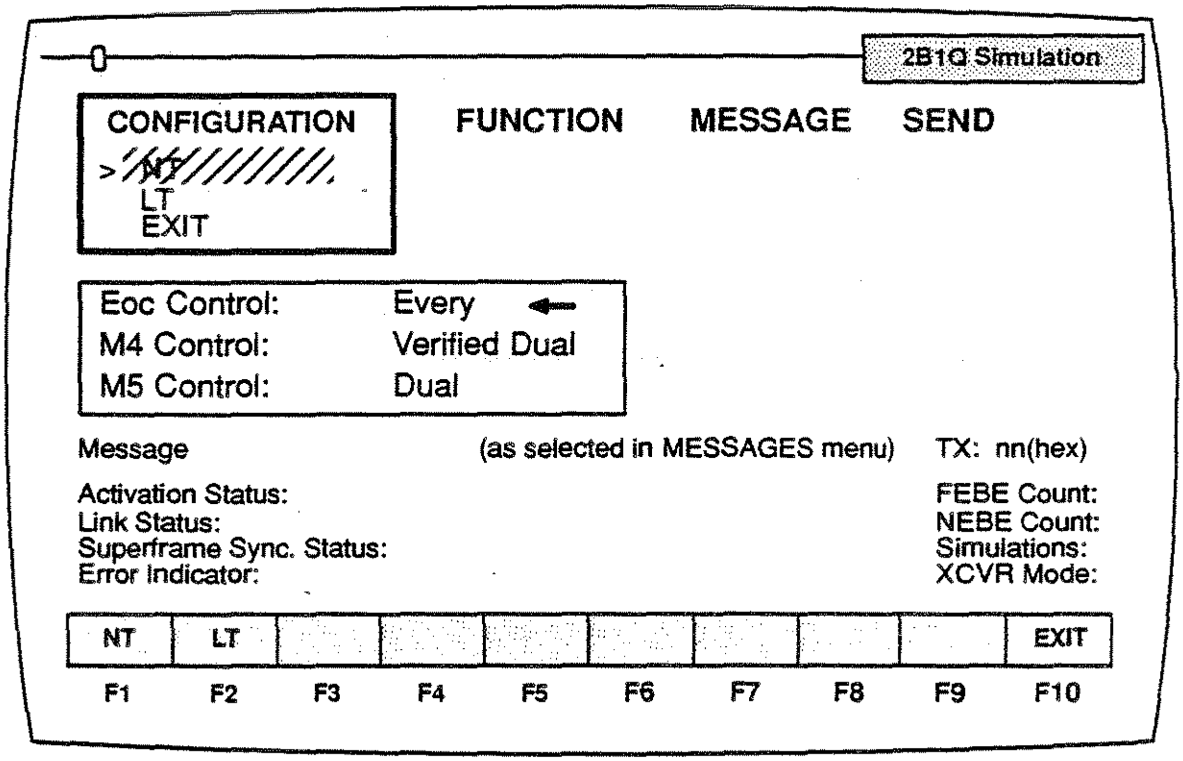
Channel Selection Assigns the channel for which the analog device is to be the interface. The channel you assign here cannot also be assigned to Port B and/or Port A

Encoding Sets the analog interface to be encoded in either A–Law or u–Law.

Idle Pattern Destination Assigns the channel to which the idle pattern is to be transmitted. Also de–activates idle pattern transmission altogether.

Bit Pattern Enter the bit pattern you want to use as the idle Pattern. This will then be transmitted to the destination channel selected above.

2B1Q SIMULATION CONFIGURATION



Eoc Control: Every
M4 Control: Verified Dual
M5 Control: Dual

Message (as selected in MESSAGES menu) TX: nn(hex)

Activation Status:
Link Status:
Superframe Sync. Status:
Error Indicator:

FEBE Count:
NEBE Count:
Simulations:
XCVR Mode:

The 10 link status messages at the bottom of this screen are read–only. For detailed explanations, see your *Protocol Interpretation Manual*, page 20–18.

Configuration:

Select the network entity you want your Chameleon to emulate:

NT = terminal device in a network.

LT = network.

Or, select EXIT to back out of the 2B1Q simulation application to the Applications Selections Menu.

After selecting either NT or LT and pressing ***RETURN***, the applicable sub–menu appears.

Select EOC Control, M4 Control, or M5 Control.

Use the Space Bar to toggle to the Control option you want.

For EOC, the options are:

- Every
- Trinal–Check, and
- (for the NT configuration only) Auto EOC Processor.

For M4, the options are:

- Verified Dual Consecutive
- Dual Consecutive
- Delta
- Every

For M45/M6, the options are:

- Dual Consecutive
- Delta
- Every

2B1Q SIMULATION CONFIGURATION (continued)

Press GO to close the sub–menu and return to the Configuration menu.

Press the right arrow to select the FUNCTION menu.

Function:

Select the desired function for the U transceiver of your Chameleon:

Activate = Local U transceiver will initiate start–up, notify remote U transceiver that it is ready to communicate over Layer 2.

Deactivate = Turns the local U transceiver off.

2B + D Loopback = Sets the local U transceiver to loop B1, B2 and D channels back to remote U transceiver.

B1 Loopback = Sets local U transceiver to loop B1 channel only back to remote U transceiver.

B2 Loopback = Sets local U transceiver to loop B2 channel only back to remote U transceiver.

Corrupt CRC = Sets local U transceiver to generate a corrupted CRC.

Return to Normal = Resets local EOC processor to initial state: terminates all outstanding EOC–controlled operations.

Reset XCVR = Resets local U transceiver chip. AFTER SELECTING AND EXECUTING THIS OPTION, YOU MUST RESET ALL CONTROL MODES IN THE CONFIGURATION SUB–MENUS.

Clr Err Counters = Sets to zero the local U transceiver FEBE and NEBE error counters.

Press GO to close the menu.

Press the right arrow to select the MESSAGE menu.

Message:

Select the EOC, M4 or M5/M6 message you want to build. The options available for each type of message depend upon the configuration you selected – NT or LT. Have you forgotten your Configuration selection? Look at the *Simulation* line in the Status Window. It will show the configuration you selected earlier in this procedure. For explanations of the options listed below, see your *Protocol Interpretation Manual*, page 20–32.

NT
- EOC
  - Unable to Comply
  - Hold State
- M4 message
  - ACT (activate)
  - PS1 (Power Supply, bit 1)
  - PS2 (Power Supply, bit 2)
  - NTM (NT Test Mode)
  - CSO (Cold Start Only)
  - Reserved 6
  - SAI (S/T Interface Activity)
  - Reserved 8
- M5/6 message
  - Reserved 51
  - Reserved 61

LT
- EOC
  - Operate 2B + D Loopback
  - Operate B1 Loopback
  - Operate B2 Loopback
  - Request corrupted CRC
  - Notify of corrupted CRC
  - Return to Normal
  - Hold State
- M4 message
  - ACT
  - DEA (deactivate)
  - Reserved 3
  - Reserved 4
  - Reserved 5
  - Reserved 6
  - UOA (U–Interface Only Activation)
  - AIB (Alarm Indication Bit)
- M5/6 message
  - Reserved 51
  - Reserved 61

Reserved 52	Reserved 52
FEBE	FEBE

Send

It is from this menu that you transmit the message built in the preceding Message menu. You can send this message only once each time. To send the same message repeatedly, press *Return* for each transmission.

ANALYSIS CONTROL/SHIFT KEYS

These Control and Shift keys provide special functions in the Analysis pages.

KEY	FUNCTION
A or a	For Dual Port machines, displays the Port A function key strip in History
B or b	For Dual Port machines, displays the Port B function key strip in History
Ctrl B	Switches on/off a line which separates events in the display
Ctrl C	Toggles between the Port A and Port B function key strip display (Dual Port only)
Ctrl E	Enables/disables the display of the ***Incomplete event*** message.
Ctrl N	Relevant for ISDN monitoring only. Toggles the display between the extended address in hex and the LTID or TGI byte interpretation.
Ctrl P	Activates History Print/File feature. See description below.
Ctrl Z	Protocol specific to SS7. Invokes the User Parts Editor.

HISTORY PRINT/FILE FEATURE

Ctrl P invokes the History Print feature which outputs a range of events to a printer or ASCII file. **Note that a file saved in this manner cannot be replayed in Analysis.** There are two ways to use this feature:

Method 1 – Enter a specified range of events:

1. Make the History page active.
2. Press *Ctrl P*. You are prompted for a file name and a range of events.
3. To output events to a printer, press *Return* when prompted for a file name. Your printer should already be connected and the Chameleon printer configuration set up.
   To output events to an ASCII file, enter a file name and press *Return*. The file will be saved to the hard disk in the following directory: A:\TEKELEC\DATA\HIST.
4. Enter the numbers of the first and last events you want to output.
5. Press *Go* to start the printer/file output.
6. To abort this function at any time, press *Cancel*.

Method 2: Highlight a range of events:

1. Make the History page active.
2. Use *Scroll*↑ or *Scroll*↓ key to position the first event you want to output at the top of the page. Press the left bracket key ([ ). This marks (highlights) the first event.
3. Use the *Scroll*↑ or *Scroll*↓ key to display the last event you want to output at the bottom of the screen. Press the right bracket key ( ]) to mark (highlight the last event).
4. Press *Ctrl P* to invoke the History Print menu.
5. To output events to a printer, press *Return* when prompted for a file name.
   To output events to an ASCII file, enter a file name and press *Return*. The file will be saved to the hard disk in the following directory: A:\TEKELEC\DATA\HIST.
6. The selected event numbers are displayed in the menu. You can change them by deleting the number and enter a new number.
7. Press *Go* to start the printer/file output. A message is displayed that indicates which events are being sent to the printer or file.
8. To abort this function at any time, press *Cancel*.

HISTORY DISPLAY KEYS

The keys and commands listed control the data that is displayed in the History page. If the selected event is not valid (for example, it was overwritten in the buffer), the first valid event following the selected event is displayed.

KEY	FUNCTION
←	The left arrow displays the oldest events in the buffer.
→	The right arrow displays the most recent events in the buffer.
↑	The up arrow scrolls the data upward continuously. Each time you press the up arrow, the scrolling speed increases. If data is scrolling downward, it decreases the speed of the downward scroll.
↓	The down arrow scrolls the data downward continuously. Each time you press the down arrow, the scrolling speed increases. If data is scrolling upward, it decreases the speed of the upward scroll.
Space bar	Stops scrolling.
Scroll ↑	The Scroll ↑ key moves data up one line each time you press the key.
Shift Scroll ↑	Shift Scroll ↑ displays the next page of data.
Scroll ↓	The Scroll ↓ key moves data down one line each time you press the key.
Shift Scroll ↓	Shift Scroll ↓ displays the previous page of data.
0 – 9	The number keys move you to a certain point in the buffer. Each number represents a percentage of the buffer, from 0% (0) to 90% (9). For example, if you press 5, the middle (50%) of the buffer is displayed.
F or f	Freeze Mode – Displays the most recent 32K of data for display on the History page. While in Freeze Mode only 32K of data can be viewed on the page; however it will not be overwritten by new data being acquired.
U or u	Un–freeze – terminates Freeze Mode and returns you to the normal History display. When unfrozen the History page displays data from the acquisition buffer.
:jump *n*	Jumps to event number n. For example, :jump 150 displays event 150 as the first event on the page. *:jump 99999* displays the end of the buffer (most recent events).
:normal	Used in conjunction with the Triggering application DISPLAY option. Selects normal triggering display mode which causes data which meets the triggering criteria to be shown in low intensity color. All other data is shown in high intensity color.
:trigger	Used in conjunction with the Triggering application DISPLAY option. Selects trigger display mode which causes only the data which meets the triggering criteria to be displayed in the History page. All other data is suppressed from the display.

DUAL LINE APPLICATION

The Dual Line application displays data in a 2–line format (DCE over DTE) which represents the actual sequence of data as it was acquired by the Chameleon. This type of display enables you to determine the overlap of data being received simultaneously from both sides of the line. To start the application, select **DUALLINE** from the Monitoring window of the Applications Selection menu. *F10* toggles between the two Dual Line modes: Run mode and Freeze mode.

Run mode causes the page to be updated as data is acquired from the line or from disk. In Run mode the display shows the following information:

- The DCE and DTE baud rates are displayed at the top of the screen.
- DCE data is displayed in brown above the DTE data
- DTE data is displayed in underlined cyan below the DCE data
- Each line displays up to 64 characters
- Interface lead states are displayed when *F3 State* is selected.
- Data is displayed in the format set selected with *F1.*
- Blank spaces between frames indicate idle time. *F2* controls the display of idle time.

The Run mode function keys are as follows:

F1 determines in what format the data is displayed: **ASCII, EBCDIC, HEX, HEXS.** If *F1* = HEXS, data is displayed in hex pairs, with pairs alternating in high and low intensity color.

F2 determines how idle data bytes are displayed. Idle data is shown as blank spaces between frames. *F2* determines how many idle data bytes are represented by each blank space. For example, if *F2* = 10, each blank space represents 10 bytes of idle data.

F3 determines what data is displayed. The options are:

Data Data is displayed, but interface lead states are not displayed.

State Both data and interface lead states are displayed.

F10 toggles between Run mode and Freeze mode.

Freeze mode freezes the Dual Line page so that it is no longer updated as data is acquired. In Freeze mode there are additional function keys which enable you to scroll through the data. The Freeze mode display is the same as the Run mode display, with the addition of these fields:

- Binary value of the DCE and DTE byte at the location of the cursor
- Hex value of the DCE and DTE byte at the location of the cursor
- ASCII or EBCDIC value of selected byte (depending on current *F1* selection)
- Time stamp indicating the time that the end of the event was acquired. The time stamp is in the format: hh:mm:ss ddd ddd (ddd ddd is the number of microseconds in decimal)

The Freeze mode function keys are the same as Run mode, with the addition of these function keys:

F7 displays the previous page of data.

F8 displays the next page of data.

F9 marks the byte at the cursor as the base line byte. When a byte is marked, the following changes occur to the Dual Line page:

- The marked byte is shown in red
- The **dtime** field displays the delta time between the marked byte and the byte at the cursor
- The bytes field displays the offset between the marked byte and the byte at the cursor

BERT APPLICATION

The Chameleon BERT application provides synchronous or asynchronous Bit–Error Rate Testing (BERT) data testing for a variety of data communications systems. The Chameleon can be configured to emulate either a DTE or a DCE over any of the Chameleon I/O modules.

When BERT is started, the BERT Setup menu appears with the following configurable parameters:

Framing selects Synchronous or Asynchronous timing.

Interface specifies whether the Chameleon will simulate a DCE or a DTE device.

Data Bits specifies the number of data bits in each byte as 8, 7, 6, or 5 bits. It is relevant only for asynchronous framing.

Stop Bits specifies the number of stop bits being used in each byte of data as 1, 1.5, or 2. It is relevant only for asynchronous framing.

Parity specifies the parity setting being used as None, Odd, or Even. It is relevant only for asynchronous framing.

Baud Rate specifies the speed (in bits per second) that the Chameleon will use to transmit or receive data. If the Chameleon is configured as a DTE using synchronous framing, the Chameleon will match the received clock.

Pattern specifies the type of data that the Chameleon will transmit or expect to receive on the line:

- Pseudo–random bit pattern of 63, 511, 2047, 4095, or 32767 bits in length
- The pattern 1010101
- The FOX message: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 1234567890 CR
- A user–defines pattern of 3 – 200 bytes in length.

Error Insertion Rate In Synchronous Framing only, sets the rate at which errors are automatically inserted into the bit stream. There are seven options available:

F1 None No automatic insertion of errors.

F2 1.04E–2 Errors inserted at the rate of 1040 in every 100,000 bits.

F3 1.02E–3 Errors inserted at the rate of 102 in every 100,000 bits.

F4 1.00E–3 Errors inserted at the rate of 100 in every 100,000 bits.

F5 9.84E–4 Errors inserted at the the rate of 98.4 in every 100,000 bits.

F6 1.00E–4 Errors inserted at the the rate of 10 in every 100,000 bits.

F7 1.00E–5 Errors inserted at the the rate of 1 in every 100,000 bits.

NOTE: You must enter the *F4* key of the BERT Setup Menu in order to activate the *F8* key of the Run–Time Menu for toggling error insertion on and off.

User Defined Preamble enables you to enter a 2–byte preamble which may be required by the remote device in order to synchronize the line.

User Preamble appears only when the User Defined Preamble parameter is YES. Enter the 2 hex bytes for your required preamble and press Return.

Block Length specifies the block length required for your testing application, in the range is 0 – 64k bits

Mode determines what the Chameleon will do during the testing session. The options are:

BERT APPLICATION (continued)

F1 REMOTE The Chameleon generates the BERT pattern and transmits it to the remote device. This device then returns the original pattern to the Chameleon, or generates a new one as transmits it back. In either case, the Chameleon does a validity check of the pattern.

F2 LOCAL The Chameleon waits to receive a BERT pattern. It then synchronizes on that pattern, checks its validity, and re–transmits the pattern to the remote device.

F3 RECEIVE The Chameleon synchronizes on a received BERT pattern and checks its validity. No pattern is generated by the Chameleon.

Duration of Test Determines how long test runs in continuous mode (see *F2 Contins*). Only indicates test duration. Enter in the format hh:mm:ss. 00:00:00 causes test to run until manually stopped (see *F3 Stop*). Maximum duration is 97 hours, 59 minutes, and 59 seconds (97:59:59).

There are two BERT run–time pages which display the statistics resulting from the Chameleon's analysis of the data received on the line. The function keys for the two BERT run–time pages are identical, as follows:

F1 1block is relevant for Remote Loopback and Local Loopback testing. It causes the Chameleon to transmit one block of data to the remote device.

F2 Contins In Remote Loopback mode, this causes the Chameleon to transmit data continuously. In Local Loopback mode, the Chameleon will begin to transmit data continuously once the line is in sync.

F3 Stop stops continuous testing mode. To continue, press *F2 Contins.*

F4 Ins Err causes the Chameleon to transmit an errored bit into the data being transmitted to the remote device.

F5 Resync causes the Chameleon to attempt to resynchronize the line.

F6 Reset resets all statistical fields in both pages to zero. In continuous mode, it resets all statistical fields and automatically resumes testing.

F7 Setup stops the test session and exits to the BERT Setup menu.

F8 Err off/on toggles the insertion of errors *ON* and *OFF*. In Synchronous Framing only, this key is activated whenever Error Insertion Rate keys *F2* through *F7* are pressed.

F9 Next toggles between the two run–time pages.

F10 Exit stops the BERT application and returns you to the Applications Selection menu.

The top of both run–time pages display identical fields. These fields are:

Elapsed Seconds displays the number of seconds which have elapsed since the test was started.

Time displays the system time as derived from the Chameleon clock.

Mode displays the current testing Mode as configured in the Setup menu.

Pattern displays the current Pattern as configured in the Setup menu.

Block Length This field displays the current Block Length.

User Preamble displays the User Preamble as configured in the Setup menu.

Status displays the testing status between the Chameleon and the remote device. It will display one of the following:

BERT APPLICATION (continued)

Idle The Chameleon is not actively performing a test.

No Sync The test is proceeding, but the line is not synchronized.

In Sync The line is synchronized and the test is proceeding.

In addition to the above fields, the first BERT run–time page displays these additional fields:

Number of Bits:

For Transmit, this field displays the total number of bits transmitted by the Chameleon to the remote device. For Receive, this field displays the total number of bits received by the Chameleon from the remote device.

Errored Bits:

For Receive, this field displays the number of errored bits received from the remote device according to the data pattern in use. For Transmit, this field displays the number of errored bits transmitted by the Chameleon to the remote device. To transmit an errored bit from the Chameleon, you must press the *F4 Ins. Err* key.

Bit Error Rate:

For Receive, this field displays the number of errored bits received since the beginning of the test session, or since the run–time display was reset using *F6 Reset*. It is calculated as the ratio of the number of bit errors to the total number of bits received. For Transmit, this field is not applicable.

Number of Blocks:

For Transmit, this field displays the total number of blocks transmitted by the Chameleon to the remote device. For Receive, this field displays the total number of blocks received by the Chameleon from the remote device.

Errored Blocks:

For Receive, this field displays the number of blocks received from the remote device with one or more bit errors. For Transmit, this field is not applicable.

Block Error Rate:

For Receive, this field displays the number of errored blocks received since the beginning of the test session, or since *F6 Reset* was pressed. For Transmit, this field is not applicable.

The second BERT run–time displays additional statistics based on the bit error rate of the received data.

Error Free Seconds:

This field displays the number of available seconds in which no bit errors have occurred on the line.

Errored Seconds:

This field displays the number of seconds in which at least one bit error has occurred.

Severely Error Seconds:

This field displays the number of seconds in which an available second has a bit error rate worse then 10E–3.

Consecutively Severely Error Seconds:

This field displays the number of consecutive seconds with bit error rates worse then 10E–3.

Degraded Minutes:

This field displays the number of degraded minutes. A degraded minute is a 60–second block of non–severely errored available seconds in which the average bit error rate, measured over the 60 seconds, is worse than 10E–6.

Unavailable Seconds:

This field displays the number of unavailable seconds. An unavailable second is a second in which the line quality is degraded enough that the Chameleon received data with more than 10 consecutive severely errored seconds.

DIRECT–TO–DISK APPLICATION

The Direct–to–Disk application stores a maximum of 30 Mbytes of traffic acquired from the line to the hard disk. Once stored to disk, traffic can be played back and analyzed off–line.

Recording Traffic with Direct–to–Disk

1. Configure the desired port for Monitoring from the line or for Simulation.
2. Press *Go* to display the Applications Selection page.
3. Move the red arrow cursor to the DIRTDSK application and press the function key that loads the application for the appropriate port.
4. Load additional applications, as desired.
5. Press *Go*. This starts the tasks that are loaded, including Direct–to–Disk. Traffic is saved in a special 30 Mbyte area of the hard disk.
6. To stop recording traffic, select the Configuration page, and stop the Direct–to–Disk application. Do not restart the Direct–to–Disk application, or it will overwrite the data that is currently in the Direct–to–Disk area of the hard disk.
7. This traffic can be replayed directly from the hard disk, or saved in a file. To record additional data to disk, first save the data that is stored in the Direct–to–Disk portion of the hard disk by following the steps below.

Saving Direct–to–Disk Data to a File

1. If necessary, stop the Direct–to–Disk or the Direct–from–Disk application. You cannot save Direct–to–Disk data if either application is running.
2. Press ***Utilities*** to invoke the Utilities menu. Select and display the Utilities menu.
3. Press *F4* ***Traffic Load/Save*** to display the Traffic Operations menu.
4. Press *F1* ***Save*** to select the Operation.
5. Enter a file name and press ***Return***. The file is saved to the hard disk unless you specify b: as part of the file name for the floppy disk drive. (If you save to a floppy disk, the maximum traffic file size is 700 Kbytes. To save more than 700 Kbytes to floppy disks, back up the Direct–to–Disk area of the hard disk using the Utilities *F8* ***Backup/Restore*** option.)
6. Press *F1* ***Direct–to–Disk*** to select the Data Source.
7. To save less than 100% of the Direct–to–Disk data, press ***Delete*** to erase the current percentage, enter the new percentage, and press ***Return***. This percentage represents the most recently recorded traffic.
8. Press *Go* and the traffic is saved with the size of the file in Kbytes displayed.
9. To replay traffic saved to a file, you must load the traffic back to the Direct–to–Disk area of the hard disk as described on the next page.

Replaying Direct–to–Disk Traffic

1. If you want to replay data currently stored in the Direct–to–Disk area of the hard disk, go to step 2.

   If you want to replay data saved to a traffic file, first load the traffic file to the Direct–to–Disk area of the hard disk, as follows:

   a. Press ***Utilities*** to invoke the Utilities menu. Select and display the Utilities menu.

   b. Press ***F4 Traffic Load/Save*** to display the Traffic Operations menu.

   c. Press ***F2 Load*** to select the Operation.

   d. Enter a name for the traffic file (including file extension) .

   e. Press ***Go*** and the file is loaded into the Direct–to–Disk area of the hard disk.

   (If you used Utilities ***F8 Backup/Restore*** to save Direct–to–Disk traffic to multiple floppy disks, use the ***F8 Backup/Restore*** to restore the data to the hard disk.)

2. Configure the Chameleon for Monitoring, selecting the appropriate protocol and port for the recorded data.

3. In the main configuration page, for the Monitoring Data Source parameter press ***F2 Disk*** to select monitoring from disk.

4. Press ***Go*** to display the Applications Selection page.

5. Load the Monitoring applications that you want to use to analyze the traffic on disk.

6. Press ***Go*** to start the monitoring applications.

7. You can now use the application pages as though you were monitoring from the line. The ***Run/Stop*** key starts and stops acquisition from the disk.

8. When the entire contents of the Direct–to–Disk area has been replayed, acquisition stops. You can replay the traffic again by selecting the Configuration page and pressing ***F6 Reset***.

STATISTICS

The Statistics application is available for the protocols listed below.

PROTOCOL	APPLICATION NAME (IN MENU)	STATISTICS PAGES AVAILABLE
BSC	BSCSTAT	BSC Line Statistics BSC CU Statistics
ISDN	Q921STAT	Q.921 Line Statistics Q.921 SAPI 0 Statistics Q.921 SAPI 16 Statistics Q.921 SAPI 63 Statistics Q.921 Other SAPI Statistics
Primary Rate Interface	PRISTAT	PRI Error Statistics
SNA	SNASTAT	SNA Session Statistics SDLC Line Statistics Session PU Statistics SNA LU Statistics SDLC PU Statistics SNA LU Line
SS#7	SS7STAT	SS7 Line Statistics
X.25	X25STAT	X.25 Line Statistics HDLC Line Statistics X.25 LCN Statistics

In addition to the Statistics data–display screen for these protocols, a Performance Page is available for X.25, SNA, SS7 and ISDN Q.921.

To display the Performance Page:

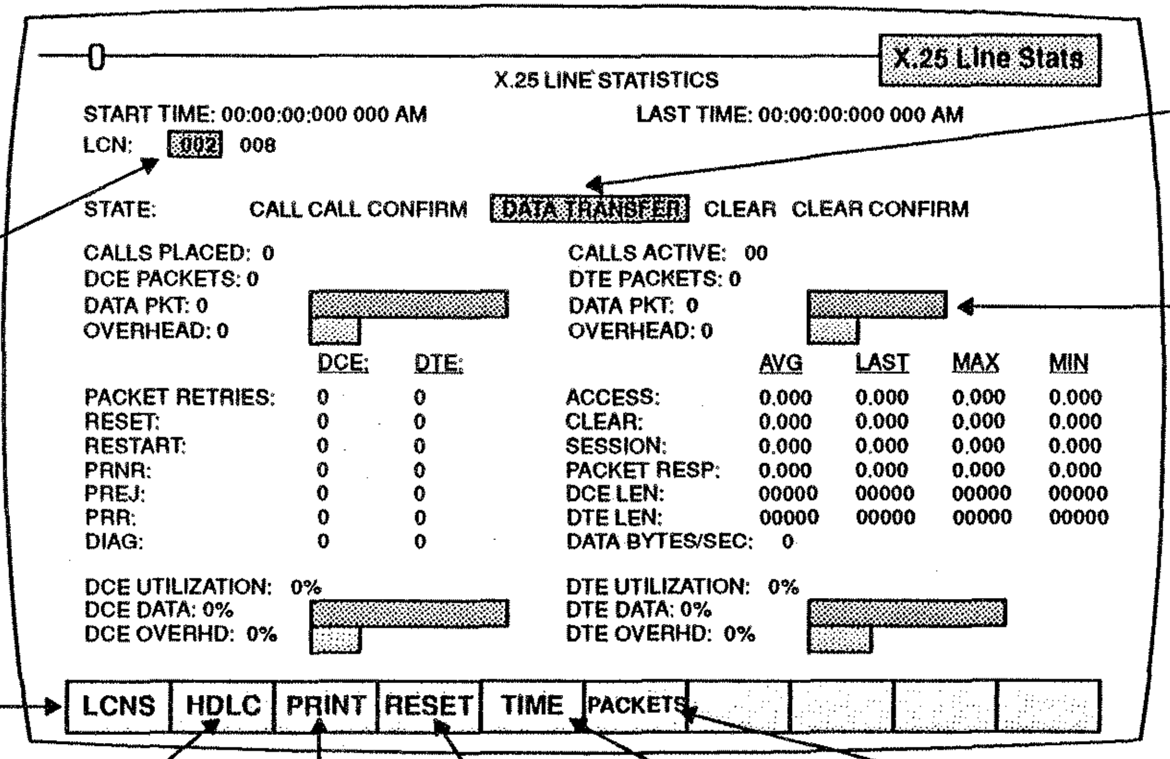
1. With the appropriate Protocol Statistics Page banner selected, press *Ctrl P.* The Performance Page banner appears on–screen.
2. Select the Performance Page banner and scroll it onto the screen, or press ***Shift Move*** ↑

To close the Performance Page:

1. With the Protocol Statistics page on–screen but NOT in blow–page mode (if it is in this mode, press ***Shift Move*** ↑ to anul that mode), select the Statistics Page banner.
2. Press *Ctrl P.* The Performance Page is closed.

STATISTICS

The function keys for all Statistics pages are similar (except in PRI statistics). A sample X.25 Statistics page with function key descriptions is provided below.



```
                              X.25 LINE STATISTICS
START TIME: 00:00:00:000 000 AM              LAST TIME: 00:00:00:000 000 AM
LCN:   002  008

STATE:         CALL CALL CONFIRM   DATA TRANSFER  CLEAR  CLEAR CONFIRM

CALLS PLACED: 0                         CALLS ACTIVE:  00
DCE PACKETS: 0                          DTE PACKETS: 0
DATA PKT: 0                             DATA PKT: 0
OVERHEAD: 0                             OVERHEAD: 0
                    DCE:     DTE:                          AVG    LAST   MAX    MIN
PACKET RETRIES:     0        0          ACCESS:            0.000  0.000  0.000  0.000
RESET:              0        0          CLEAR:             0.000  0.000  0.000  0.000
RESTART:            0        0          SESSION:           0.000  0.000  0.000  0.000
PRNR:               0        0          PACKET RESP:       0.000  0.000  0.000  0.000
PREJ:               0        0          DCE LEN:           00000  00000  00000  00000
PRR:                0        0          DTE LEN:           00000  00000  00000  00000
DIAG:               0        0          DATA BYTES/SEC:    0

DCE UTILIZATION:  0%                    DTE UTILIZATION:  0%
DCE DATA: 0%                            DTE DATA: 0%
DCE OVERHD:  0%                         DTE OVERHD:  0%
```

Current call status is highlighted.

Most recent packet received from highlighted address.

Graphic representation of statistics data

Displays addresses so that you can activate statistics pages for them.

X.25: LCNs
SNA: PUs/LUs
BSC: CUs
ISDN: SAPIs

Displays protocol layer so that you can activate statistics pages for it.

X.25: HDLC
SNA: Session

Prints a statistical report if the Chameleon is configured for, and connected to a printer.

Resets all values and timers to zero for all statistics pages for that protocol.

Displays time or date and time

Determines whether the number of PACKETS or number of BYTES is displayed for DCE/DTE Packets, Data Packets, and Overhead.

TRIGGERING APPLICATION

TRIGGER STRUCTURE

Name	Status	1–4 Conditions	1–4 Actions
1–4 character default or user–specified name	1st Time Enable to fire once Disable Whenever Enable forever	Selected by function keys. (See below and next page.)	Executed if all trigger conditions are met. Selected by function keys. (See below.)

TRIGGERING LOGIC

trg1 — Condition 1 **AND** Condition 2 **AND** Condition 3 **AND** Condition 4

OR

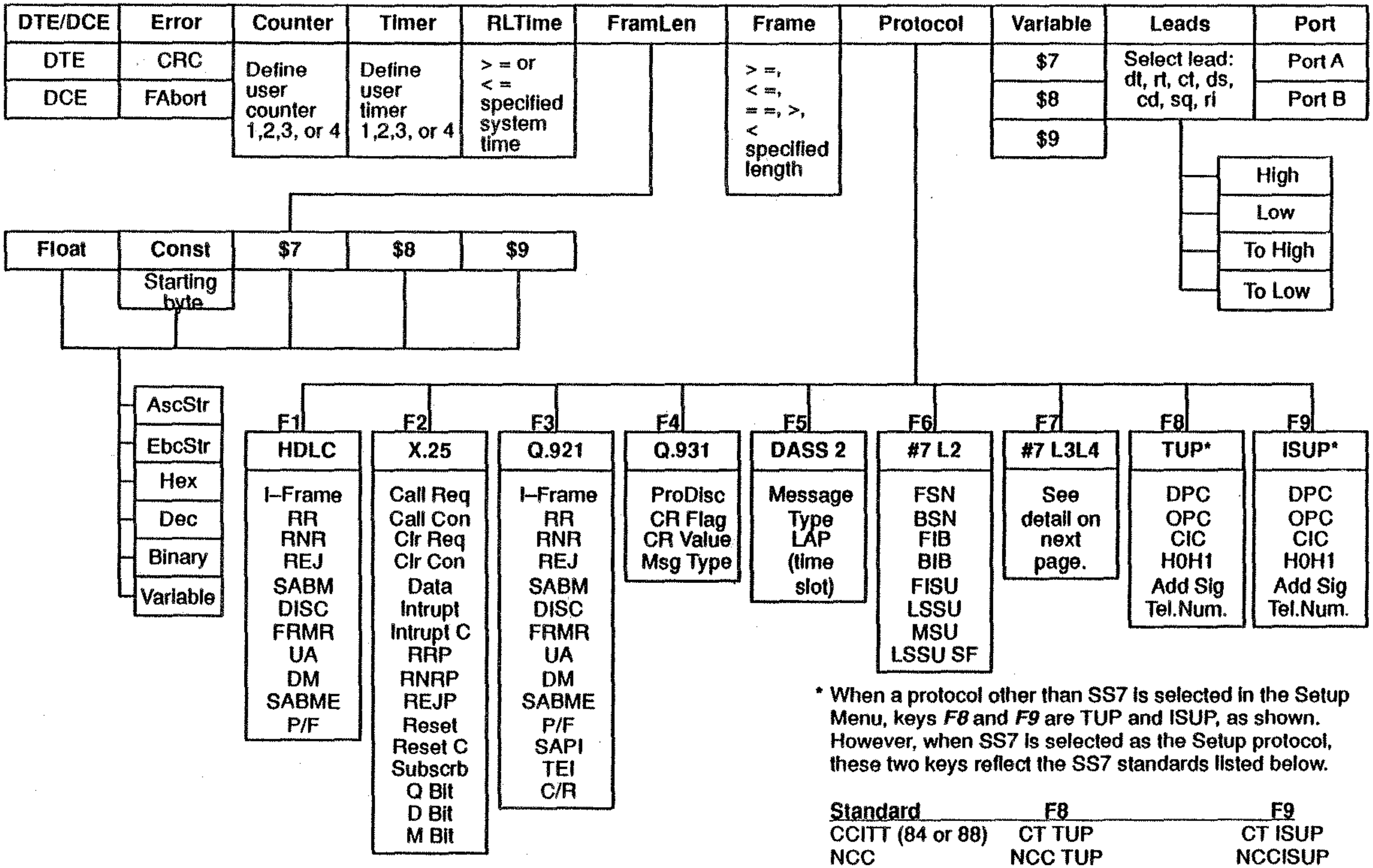
trg2 — Condition 1 **AND** Condition 2 **AND** Condition 3 **AND** Condition 4

●
●
●

CONDITIONS See next page for function key map. To use logical NOT condition, press Shift–function key.

CONDITIONS	**DTE/DCE**	Triggers on either DCE events, DTE events, or both.
	Error	Triggers on CRC and frame abort errors.
	Counter	Triggers on a user–defined counter value.
	Timer	Triggers on a user–defined timer value.
	RLTime	Triggers on Real–time Clock value.
	Frame	Triggers on user–specified data string in a frame.
	Framelen	Triggers on frame length in bytes.
	Protocol	Protocol–specific options for X.25, HDLC, Q.921, Q.931, SS#7, or DASS 2.
	Variable	Compares an integer variable with another variable or constant.
	Leads	Triggers on interface lead states or changes.
	Port	Triggers on events from Port A, Port B, or both.
ACTIONS	**Arm**	Arms (enables) another trigger.
	Stats	Processes event, prints a report, or resets the statistics application.
	Display	Displays events in the Real Time page.
	=>Disk	Records events to the Direct–To–Disk area of the hard disk.
	Mesg	Displays the message "Trigger Fired" and beeps.
	StopAcq	Stops the acquisition of traffic from the line.
	IncCnt	Increments the specified counter by one.
	ResCnt	Resets specified counter.
	Timer	Starts, stops, or resumes a specified timer.
	SetVars	Stores a value to a variable.
	V Arith	Change the value of one of the integer variables.
	TrigOut	Sets Chameleon to signal remote monitoring device upon detection of triggering event.

TRIGGERING CONDITIONS (Function Keys)

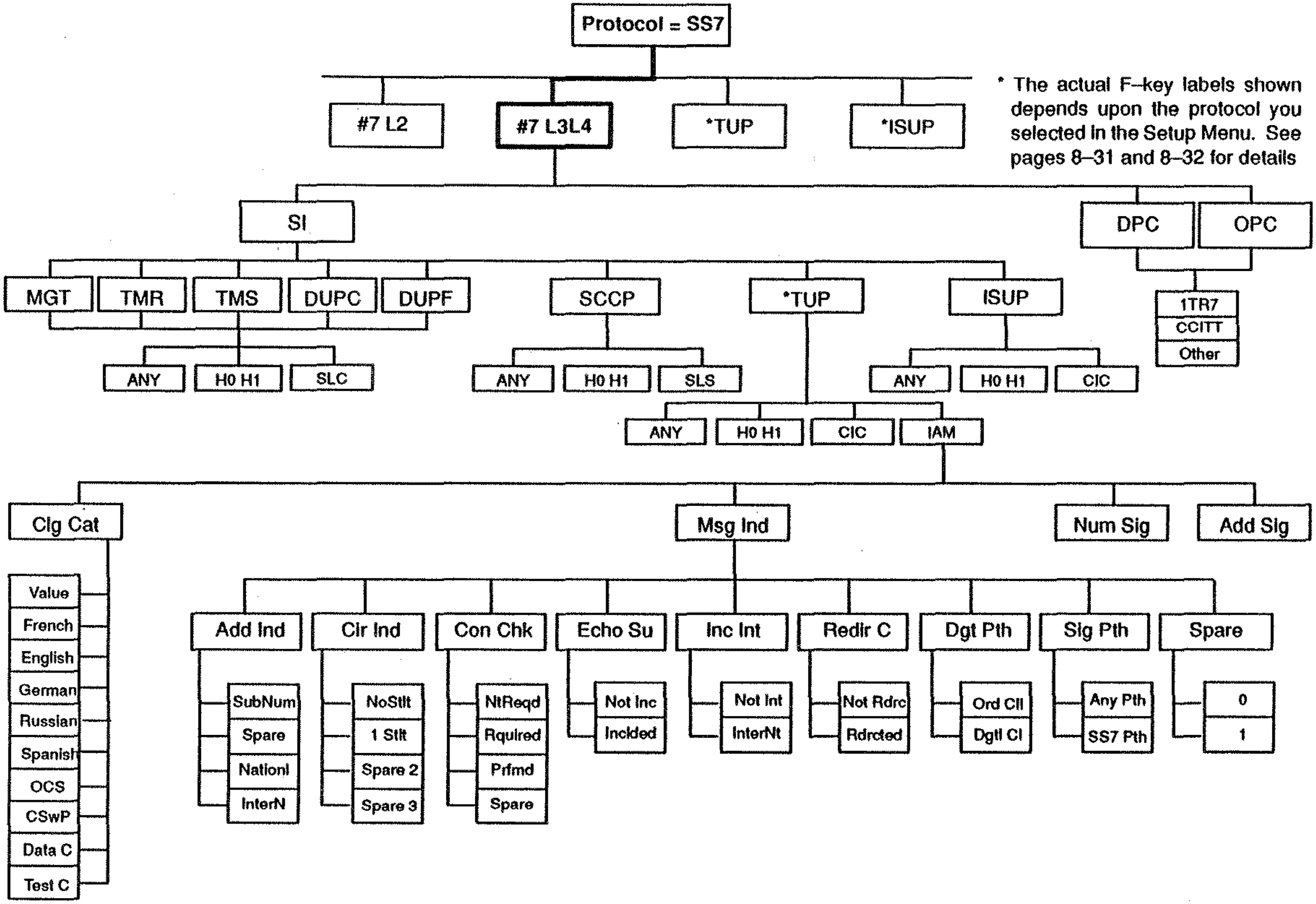


- DTE/DCE
  - DTE
  - DCE
- Error
  - CRC
  - FAbort
- Counter
  - Define user counter 1,2,3, or 4
- Timer
  - Define user timer 1,2,3, or 4
- RLTime
  - > = or < = specified system time
- FramLen
  - Float
  - Const
    - Starting byte
  - $7
  - $8
  - $9
    - AscStr
    - EbcStr
    - Hex
    - Dec
    - Binary
    - Variable
- Frame
  - > =, < =, = =, >, < specified length
- Protocol
  - F1 HDLC: I–Frame, RR, RNR, REJ, SABM, DISC, FRMR, UA, DM, SABME, P/F
  - F2 X.25: Call Req, Call Con, Clr Req, Clr Con, Data, Intrupt, Intrupt C, RRP, RNRP, REJP, Reset, Reset C, Subscrb, Q Bit, D Bit, M Bit
  - F3 Q.921: I–Frame, RR, RNR, REJ, SABM, DISC, FRMR, UA, DM, SABME, P/F, SAPI, TEI, C/R
  - F4 Q.931: ProDisc, CR Flag, CR Value, Msg Type
  - F5 DASS 2: Message Type, LAP (time slot)
  - F6 #7 L2: FSN, BSN, FIB, BIB, FISU, LSSU, MSU, LSSU SF
  - F7 #7 L3L4: See detail on next page.
  - F8 TUP*: DPC, OPC, CIC, H0H1, Add Sig, Tel.Num.
  - F9 ISUP*: DPC, OPC, CIC, H0H1, Add Sig, Tel.Num.
- Variable
  - $7
  - $8
  - $9
- Leads
  - Select lead: dt, rt, ct, ds, cd, sq, ri
    - High
    - Low
    - To High
    - To Low
- Port
  - Port A
  - Port B

* When a protocol other than SS7 is selected in the Setup Menu, keys *F8* and *F9* are TUP and ISUP, as shown. However, when SS7 is selected as the Setup protocol, these two keys reflect the SS7 standards listed below.

Standard	F8	F9
CCITT (84 or 88)	CT TUP	CT ISUP
NCC	NCC TUP	NCCISUP
NTT	NTT TUP	NTTISUP
ANSI	ASN TUP	ANSISUP
1TR7	TR7 TUP	TR7ISUP

SS#7 LEVEL 3 AND LEVEL 4 TRIGGERING OPTIONS



* The actual F–key labels shown depends upon the protocol you selected in the Setup Menu. See pages 8–31 and 8–32 for details

SIMULATOR ROAD MAP

If you are here:	NEXT STEP	
	And you want to:	Do this:
FRAMEM, SIMP/L BSC, or Async simulation prompt!	Stop the simulator	Enter: MENU <RETURN> OR Access the Configuration page, move the red arrow cursor to the simulator name, and press the function key that stops the simulator on the desired port.
	Access the parameter set–up menu	Enter: SETUP <RETURN>
	Write programs	Read the Chameleon 32 Simulation Manual (Chapter 3) for Chameleon BASIC fundamentals AND read the chapter that describes your simulation language.
SITREX SIMULATOR ACTIVE	Return to the main menu	Enter: PS <RETURN> HALT <RETURN>
	Access the parameter set–up menu	Enter: PS <RETURN> HALT <RETURN> Press: F2 GO
	Enter command mode (!) Exit command mode	Enter: PS <RETURN> Enter: EXIT <RETURN>
	Activate the trace buffer Deactivate the trace buffer	Enter: PP <RETURN> Enter: CTRL P
Any Parameter Set–Up Menu	Access the simulation prompt (!) to write programs	Press: Z
	Return to the main menu	Press: ESC
	Change parameter values	Read the Chameleon 32 Simulation Manual (Chapter 2.2) for general information about the set–up menus AND read the chapter that describes the menu for the simulation language you are using.
	Save parameter values	Press: S

FRAMEM LAPD DEFAULT MNEMONIC TABLE

The I–field column in the table indicates whether the mnemonic can have an I–field. If an I–field is permitted (using the DEFINE command), the letter I appears in the I–Field column.

MNEMONIC	I–FIELD	DECIMAL	HEX	BINARY
IFRAME		0	00	00000000
SNRME		207	CF	11001111
SARME		79	4F	01001111
SABME		111	6F	01101111
SREJ		13	0D	00001101
SNRM		131	83	10000011
SARM		15	0F	00001111
SABM		47	2F	00101111
DISC		67	43	01000011
RSET		143	8F	10001111
FRMR		135	87	10000111
TEST		227	E3	11100011
CMDR		135	87	10000111
RNR		5	05	00000101
REJ		9	09	00001001
SIM		7	07	00000111
XID		175	AF	10101111
RIM		7	07	00000111
NSI		3	03	00000011
RQI		7	07	00000111
ROL		15	0F	00001111
NSP		35	23	00100011
RR		1	01	00000001
UI		3	03	00000011
UP		35	23	00100011
DM		15	0F	00001111
UA		99	63	01100011
RD		67	43	01000011

FRAMEM HDLC/SDLC MNEMONIC TABLE

The DEFSUB column can reference a line number. If this type of frame is received, program control jumps to the program line number specified in this column and executes the subroutine. Refer to the FRAMEM DEFSUB command for more information.

MNEMONIC	DECIMAL	HEX	BINARY	DEFSUB
IFRAME	0	00	00000000	
SNRME	207	CF	11001111	
SARME	79	4F	01001111	
SABME	111	6F	01101111	
SREJ	13	0D	00001101	
SNRM	131	83	10000011	
SARM	15	0F	00001111	
SABM	47	2F	00101111	
DISC	67	43	01000011	
RSET	143	8F	10001111	
FRMR	135	87	10000111	
TEST	227	E3	11100011	
CMDR	135	87	10000111	
RNR	5	05	00000101	
REJ	9	09	00001001	
SIM	7	07	00000111	
XID	175	AF	10101111	
RIM	7	07	00000111	
NSI	3	03	00000011	
RQI	7	07	00000111	
ROL	15	0F	00001111	
NSP	35	23	00100011	
RR	1	01	00000001	
UI	3	03	00000011	
UP	35	23	00100011	
DM	15	0F	00001111	
UA	99	63	01100011	
RD	67	43	01000011	

SIMP/L LAPD DEFAULT MNEMONIC TABLE

MNEMONIC	FIELD WIDTH (BITS)	DEFINITION/ Q.931 MESSAGE OCTET
MESTYP	7	Message type/fourth octet
SHFTID	3	Shift 10/fourth octet (Shift info. element)
LOKBIT	1	Shift lock bit/fourth octet (Shift info. element)
CODSET	3	Code set/fourth octet (Shift info. element)
CRLEN	4	Call reference length/second octet
CREF7	7	Call reference/third octet
CREF8	8	Call reference/third octet
NOEXT	1	No extended bit/fourth octet filler
PDIS	8	Protocol discriminator/first octet
FIL4	4	Four bit filler/second octet
PAD1	1	One bit filler/fourth octet filler
PAD2	2	Two bit filler
EXT	1	Extend bit
RI	16	Reference number/TEI field
AI	7	Action indicator/TEI field

SIMP/L HDLC DEFAULT MNEMONIC TABLE

MNEMONIC	FIELD WIDTH (BITS)
LCGN	4
PKID	8
P(S)	3
P(R)	3
PAD1	1
MBIT	1
DBIT	1
QBIT	1
PAD2	2
GFI	4
LCN	8

BSC DEFAULT MNEMONIC TABLE

MNEMONIC	DECIMAL	HEX	BINARY
ACK0	112	70	01110000
ACK1	97	61	01100001
WABT	127	7F	01111111
SOH	1	01	00000001
STX	2	02	00000010
ETB	38	26	00100110
ETX	3	03	00000011
ITB	31	1F	00011111
EOT	55	37	00110111
ENQ	45	2D	00101101
DLE	16	10	00010000
SYN	50	32	00110010
ACK	46	2E	00101110
NAK	61	3D	00111101

ASYNC DEFAULT MNEMONIC TABLE

MNEMONIC	DECIMAL	HEX	BINARY
SPACE	32	20	00100000
BELL	7	07	00000111
NULL	0	00	00000000
SOH	1	01	00000001
STX	2	02	00000010
ETX	3	03	00000011
EOT	4	04	00000100
ENQ	5	05	00000101
ACK	6	06	00000110
DLE	16	10	00010000
DC1	17	11	00010001
DC2	18	12	00010010
DC3	19	13	00010011
DC4	20	14	00010100
NAK	21	15	00010101
SYN	22	16	00010110
ETB	23	17	00010111
CAN	24	18	00011000
SUB	26	1A	00011010
ESC	27	1B	00011011
DEL	127	7F	01111111
BS	8	08	00001000
HT	9	09	00001001
LF	10	0A	00001010
VT	11	0B	00001011
FF	12	0C	00001100
CR	13	0D	00001101
SO	14	0E	00001110
SI	15	0F	00001111
EM	25	19	00011001
FS	28	1C	00011100
GS	29	1D	00011101
RS	30	1E	00011110
US	31	1F	00011111

BASIC COMMANDS

@ References the array.
@(exp)   exp =array subscript

ABS Returns the absolute value of an integer or numeric variable (integer).
ABS(x)

ASC$ EBCDIC to ASCII conversion.
ASC$($x)

ATIME$ Returns the ASCII value of the realtime stamp.
ATIME$

AUTO Automatic line numbering. Start at 10, increment by 10.
AUTO x   Start at x, increment by 10.
AUTO x,y   Start at x, increment by y.

BCD$ ASCII to BCD conversion.
BCD$($x)

BLK Display blinking text.
BLK

BLKHLF
Display blinking text in double intensity.
BLKHLF

BLKREV
Display blinking text in reverse video.
BLKREV

BLKUND
Display blinking underlined text .
BLKUND

CALL Calls a program file as a subroutine.
CALL "filename"

CHAIN Loads and runs a program file.
CHAIN"filename"

CHR$ Assigns the binary equivalent of an ASCII value.
CHR$(exp)

CLEAR Clears the trace buffer.
CLEAR

CLOSE Closes all open files.
CLOSE I   Closes input file only.
CLOSE O   Close output and append files only.

CLS Clears the screen of text.
CLS

COUPLER
Configures the Chameleon 32 hardware to transmit and receive frames using a parameter file.
COUPLER "filename"

DEC$ Converts a numeric expression into a string of ASCII decimal characters.
$X = DEC$(exp)

DEFINE Defines a mnemonic for the mnemonic table. Syntax is protocol–specific.

DEL Deletes a line from the screen.
DEL

DELETE Deletes a mnemonic from the table.
DELETE "name"

DISPF Displays the last frame transmitted or received. Not available in Async. SIMP/L uses the RDISPF (received) and TDISPF (transmitted) commands.
DISPF

EBC$ ASCII to EBCDIC conversion.
$A = EBC$($B)

EDIT Edits a line from the program in memory using commands below.
EDIT x   x = line number
Move cursor 1 space left.
Displays next character.
CTRL P   Displays the entire line to the right of the cursor.
←
→
CTRL X   Erases the entire line, including line number.
CTRL D   Deletes the next *un–displayed* character from memory.
CTRL I   Inserts a space.
RETURN   Saves the line to the left of the cursor.
CTRL Z   Exits edit mode without saving changes.

EOF Read–only variable that indicates if end of data file is reached.
EOF = 0
Not EOF
EOF = 1 EOF reached
PRINT EOF
IF EOF...

ERAEOL Erases text to the end of the line.
ERAEOL

ERAEOS Erases text to the end of the screen.
ERAEOS

ERASE Delete lines from program in memory.
ERASE x,y
x = first line number
y = last line number

EXIT Returns control to a calling program from a program that has been CALLed.
EXIT

FDEFINE Defines the function key assignments.
FDEFINE KEYx=~statement^~
x =function key (1 – 10)
~ marks beginning and end
^=carriage return between statements

BASIC COMMANDS

FILES Lists the files on a specified disk drive.
FILES A Lists files on hard disk
FILES B Lists files on floppy disk

FLIST Lists the ten function key assignments
FLIST

FLOAD Loads a function key definition file into memory.
FLOAD "filename"

FLUSH Clears the acquisition buffer.
FLUSH

FOR Controls looping in programs. Must be used with NEXT
FOR x=exp1 TO exp2 [STEP exp3]

NEXT x

x is a numeric variable
exp1 is the beginning value of x
exp2 is the maximum value of x
exp3 is the step increment

FREE Read–only variable that returns the number of free mnemonic table entries.
PRINT FREE

IF FREE...

FSAVE Saves function key assignments.
FSAVE "filename"

GOSUB Sends program to a specific line number to execute a subroutine.
GOSUB exp
exp = line number

GOTO Sends program control to a specific line number.
GOTO exp
exp = line number

HEX$ Creates an ASCII 4–character string which is the HEX equivalent of exp.
$A = HEX$(exp)

HEX Assigns a string variable value in hexadecimal.
$A = HEX>exp

HLF Causes text to be displayed in double intensity (highlight)
HLF

HLFUND Displays text in double intensity and underlined.
HLFUND

IF Allows program flow to be changed based on a decision.
IF x op y command

x and y are numeric variables
op is a logical or arithmetic operator
command is the command to execute if the statement is true

INKEY$ Assigns the next character typed on the keyboard to a string variable.
$A=INKEY$

INPUT Stores keyboard input in a variable.
INPUT "prompt",x

prompt is the text that you want displayed (optional)
x is the variable that stores the keyboard input
, displays the variable name (optional)

$INPUT Assign a string variable from the keyboard.
$INPUT $A

INS Inserts a blank line on the screen.
INS

INSTR Returns the offset (position) of a substring within the main string.
x = INSTR(str1,str2)
str1=main string
str2= substring.

KILL Deletes a file from disk.
KILL"filename",x

x is the file type:

P	Program
T	Trace
M	Mnemonic table
D	Data
S	Setup (parameter)
F	Function key definition
A	All types

LEFT$ Assigns a specified number of characters from the left end of one string to another string.
$A = LEFT$($x,exp)
exp = number of characters from the left end of $x

LEN Assigns the length of a string variable to a numeric variable.
A= LEN($x)

$x is a string variable
A is the numeric variable

LET Assigns values to numeric or string variables.

LET x = exp	Numeric variable
LET $A = "xxx"	String variable

LFILES Outputs file directory to printer.

LFILES A	Prints hard disk directory
LFILES B	Prints floppy directory

LFLIST Outputs current function key assignments to a printer or remote device.
LFLIST

BASIC COMMANDS

LIST Displays program in memory.
Lists entire program.
LIST x — Lists program from line x to end.
LIST x,y — Lists program from line x to line y.
LIST ,y — Lists program from beginning to line y.

LLIST Outputs program in memory to a printer.
LLIST — Prints entire program.
LLIST x — Prints program from line x to end.
LLIST x,y — Prints program from line x to line y.
LLIST ,y — Prints program from beginning to line y.
LMLIST — Outputs the mnemonic table in memory to a printer.
LMLIST

LOAD Loads a program file into memory.
LOAD "filename"

LTPRINT
Outputs the contents of the trace buffer to a printer.
LTPRINT

MENU Exits the simulator and returns to the main menu.
MENU

MERGE Combines a program file with the program in memory.
MERGE"filename"

MID$ Assigns characters from the middle of a string to a string variable.
$A = MID$($x,exp1,exp2)
exp1 is the position of the first character
exp2 is the number of characters

MLIST Displays the mnemonic table in memory.
MLIST

MLOAD Loads a mnemonic table into memory.
MLOAD"filename"

MSAVE Saves the mnemonic table in memory to disk.
MSAVE"filename"

NEW Deletes the program in memory.
NEW

NEXT Increments the counter in a FOR loop.
NEXT x

NRM Cancels display effects commands (blinking, underline, double intensity).
NRM

OPEN Opens a data file.
OPEN "I","filename" — Opens a file for input.
OPEN "O","filename" — Opens a new file for output.
OPEN "A","filename" — Opens file for output to the end of the file.

PRINT Displays a string, expression, or variable.
PRINT"string" — Prints the string.
PRINT $A — Prints string variable.
PRINT x — Prints numeric variable.
PRINT %x — Prints x in hex.
Options:
, acts as a field separator
\)suppresses a line feed
; suppresses the carriage return

READ Reads next record from an input file.
READ $A

REC Protocol–specific command that transfers data from the acquistion buffer to the trace buffer.

REM Programmer's internal remark.
REM comment

RESEQ Re–numbers the line numbers of the program in memory.
RESEQ — Start at line 10, increment by 10.
RESEQ {EXPR1} — Start at x, increment by 10.
RESEQ {EXPR1} {EXPR2} — Start at x, increment by y.

RETURN
Returns program control from a subroutine called by a GOSUB.
RETURN

REV Displays text in reverse video.
REV

REVHLF
Displays text in reverse video in double intensity.
REVHLF

REVUND
Displays text in reverse video and underlined.
REVUND

RIGHT$ Assigns a specified number of characters from the right end of one string to another string.
$A = RIGHT$($x,exp)
$x is the string
exp defines the number of characters from the right

RND Returns a random number.
RND(x)

BASIC COMMANDS

RUN Executes the program in memory.
RUN

SAVE Saves the program in memory to disk.
SAVE "filename"

SET Sets physical interface signal to 1 or 0. Not available in SIMP/L or FRAMEM DMI.
SET xxx = y y is a 1 or 0
xxx is one of the following:
CTS (DCE)
DSR (DCE)
DCD (DCE)
RI (DCE)
SDCD (DCE)
DTR (DTE)
RTS (DTE)

SETUP Accesses the parameter set–up menu. Not available in FRAMEM DMI.
SETUP

SIZE Returns size of free program area in bytes.
PRINT SIZE
IF SIZE...

STOP Terminates program execution.
STOP

TEST Tests an interface signal for 1 or 0.
TEST xxx = y command
y is 1 or 0
xxx is one of the following:
CTS DTE)
DSR (DTE)
DCD (DTE)
RI (DTE)
SDCD (DTE)
DTR (DCE)
RTS (DCE)
command is the command to execute when the TEST condition is true.

TFREE Returns the length of the unused trace buffer in bytes.
PRINT TFREE
IF TFREE...

TIM0 Timer which counts down in ten millisecond (.01) intervals
TIM0 = x

TIM1 Timer which counts up in ten millisecond (.01) intervals
TIM1 = x

TIM2 Timer which counts down in seconds
TIM2 = x

TIM3 Timer which counts up in seconds
TIM3 = x

TIME Read–only variable that returns a specified byte of the system time in BCD digits.
TIME(x)
x is in the range 0 to 4 and specifies the unit of time, as follows:
0 – hours
1 – minutes
2 – seconds
3 – 1/100s seconds (.01)
4 – 10s of milliseconds (.0001)

TIME$ Assigns the current time to a string variable.
TIME$

TLOAD Loads a trace file into memory.
TLOAD"filename"

TPRINT Displays the contents of the trace buffer.
TPRINT

TROFF Turns off the program trace facility (debug mode).
TROFF

TRON Turns on the program trace facility (debug mode).
TRON

TSAVE Saves the contents of the trace buffer to a trace file.
TSAVE "filename"

UND Displays text in underline.
UND
PRINT UND *text*

VAL Converts a numeric ASCII string to its integer form.
A= VAL($A)

WRITE Writes a string variable to a data file opened for output.
WRITE $A

XYPLOT
Moves the cursor to a specified postion on the screen.
XYPLOT(y,x)
y = y–coordinate (row), range 0 – 21
x = x–coordinate (column), range 0 – 79

FRAMEM COMMANDS

ABORTRAN
Transmits a frame with an abort sequence. The frame must be greater than 4 bytes in length.
ABORTRAN

BADTRAN
Transmits a frame with a bad CRC. . The frame must be greater than 4 bytes in length.
BADTRAN
BADTRAN $A

CRC Indicates if received frame had a good or bad CRC.
CRC=0=Good CRC
CRC=1=Bad CRC
PRINT CRC
IF CRC...

DEFINE Defines new mnemonics or redefines existing mnemonics.
DEFINE "NAME",I=x LAPD
DEFINE"name"=x All others

name =mnemonic name
x is a numeric expression
I=I–field permission (LAPD)

DEFSUB
Defines the line number to jump to when the received frame matches a specific mnemonic.
DEFSUB"NAME"=xxxx
name=defined mnemonic
xxxx=line number of program to execute if that mnemonic is received

EXTEND
Selects extended mode addressing.
EXTEND

GET Gets two bytes (low byte, high byte) from an I–field.
x=GET exp
PRINT GET exp

MOD Specifies modulo 8 or modulo 128 sequencing.
MOD8
MOD128

NORM Selects normal mode addressing.
NORM

PUT Defines a specified byte in an I–field for transmission.
PUT exp1,exp2
exp1=byte no. from start of I–field
exp2=value assigned to that byte

REC Assigns the next received frame in sequence from the acquisition buffer and to 0 or more string variables.
REC $A, $B...

RXADDR
Address field of the received frame.
PRINT RXADDR
IF RXADDR...

RXC/R C/R bit extracted from an FRMR field.
PRINT RXC/R
IF RXC/R...

RXDIAG Last byte of an FRMR (WXYZ bits)
PRINT RXDIAG
IF RXDIAG...

RXFCTL Control field of the received frame without the poll/final bit, N(S), and N(R).
PRINT RXFCTL

RXFRLEN
Length of the received frame.
PRINT RXFRLEN
IF RXFRLEN...

RXN(R) N(R) of the received frame, if a supervisory frame or an I–frame.
PRINT RXN(R)

RXN(S) N(S) of the received frame, if a supervisory frame or an I–frame.
PRINT RXN(S)

RXP/F Poll/final bit of the received frame.
PRINT RXP/F
IF RXP/F...

RXRFCTL
Rejected frame control field of the received frame.
PRINT RXRFCTL
IF RXRFCTL...

RXRP/F Poll/final bit of a received rejected frame control field.
PRINT RXRP/F
IF PXPR/F

RXV(R) V(R) of the rejecting station for a rejected frame.
PRINT RXV(R)
IF RXV(R)...

RXV(S) V(S) of the rejecting station for a rejected frame.
PRINT RXV(S)
IF RXV(S)...

STATUS Displays the current addressing mode and modulo.
STATUS

TPRINT Displays the contents of the trace buffer.
TPRINT

FRAMEM COMMANDS

TRAN Transmits a frame with a good CRC.
TRAN
TRAN $A

TXADDR
Sets the value of the address field of the frame being transmitted.
TXADDR = xx

TXC/R Sets the value of the C/R bit of the FRMR frame being transmitted.
TXC/R = 1
TXC/R = 0

TXDIAG Sets the value of the last byte (WXYZ bits) of an FRMR field.
TXDIAG = &xx
xx is a 2–digit hex value

TXFCTL Sets the value of the frame control field of the frame being transmitted.
TXFCTL = udm
udm = user–defined mnemonic

TXIFIELD
Sets or adds to the contents of an I–field for a frame being transmitted.
TXIFIELD = $A Sets I–field to $A.
TXIFIELD+$A Adds $A to I–field
TXIFIELD = HEX>ABCD
TXIFIELD = ASC>ABCD
TXIFIELD = EBC>ABCD
TXIFIELD + HEX>0D0A
TXIFIELD + ASC>ABCD
TXIFIELD + EBC>ABCD

TXN(R) Sets the value of N(R) of the frame being transmitted.
TXN(R) = x

TXN(S) Sets the value of N(S) of the frame being transmitted.
TXN(S) = x

TXP/F Sets the poll/final bit of the frame being transmitted.
TXP/F = x

TXRFCTL
Sets the rejected frame control field of a frame being transmitted.
TXRFCTL = RXFCTL

TXRP/F Sets the poll/final bit of a rejected frame control field for the frame being transmitted.
TXRP/F = RXRP/F

TXV(R) Sets the value of V(R) for the frame being transmitted.
TXV(R) = TXN(R)

TXV(S) Sets the value of V(S) for the frame being transmitted.
TXV(S) = TXN(S)

FRAMEM LAPD COMMANDS AND VARIABLES

FILL Changes the interframe fill pattern.
FILL=FF
FILL=7E

RXCR C/R bit of the received frame.
PRINT RXCR
IF RXCR...

RXSAPI SAPI of the received frame.
PRINT RXSAPI
IF RXSAPI

RXTEI TEI of the received frame.
PRINT RXTEI
IF RXTEI...

TXCR Sets the value of the C/R bit of the frame being transmitted.
TXCR=1
TXCR=0

TXSAPI Sets the value of the SAPI for the frame being transmitted.
TXSAPI = x

TXTEI Sets the TEI for the frame being transmitted.
TXTEI = x

FRAMEM DMI COMMANDS

FRAMEM DMI COMMANDS AND VARIABLES

CAUSE Returns cause of an on–hook state.
PRINT CAUSE
IF CAUSE...
CAUSE has the following values:
0 Start of simulation
1 Call rejected, no match on address digits
2 No wink received before T1 timeout
3 No off–hook before T7 timeout
4 Local disconnect
5 Remote disconnect

CHADMIN Defines the call setup mode to use.
CHADMIN = x
x is one of the following:
0 = Wink–start in/Wink–start out
1 = Auto–start in/Wink–start out
2 = Wink–start in/Auto–start out
3 = Auto–start in/Auto–start out

CONNECT Performs call setup procedures and seizes the channel selected on the TE820A.
CONNECT

DCALLED Displays the phone number to be outpulsed (dialed)
DCALLED

DCALLING Displays the numbers inpulsed (received).
DCALLING

DISCONNECT Causes the Chameleon 32/TE820A to go on–hook.
DISCONNECT

DMATCH Displays the number the Chameleon 32 will accept for incoming calls in Wink mode.
DMATCH

DTIMERS Displays the current timer settings.
DTIMERS Displays timer settings.
Tx=yy Sets timers.
x is the timer number, range 1 – 8
yy is the value, range 1 – 99

T1 Time between an incoming seizure and the start of the outgoing wink.
T2 Duration of the wink signal.
T3 Time between the end of the wink signal and the first dial pulse.
T4 The duration of a break pulse.
T5 The duration of a make pulse.
T6 Inter–digit time
T7 Dialing timeout
T8 Minimum disconnect time.

FRAMEM DMI COMMANDS AND VARIABLES

GLARE Indicates if a glare condition exists.
GLARE = 0=No glare condition.
GLARE = 1=Glare condition
PRINT GLARE
IF GLARE...

MATCH Specifies which incoming calls will be accepted.
MATCH="x"
x is a valid incoming number.

OUTNUM Sets the number to be outpulsed (dialed).
OUTNUM = "x"
x is the phone number, maximum of 30 digits

RESET Clears the acquisition buffer and resets the state of the call to its start of simulation disconnected state.
RESET

RESPTIME Indicates how busy your switch is.
PRINT RESPTIME
IF RESPTIME...

STATE Returns the state of a call and the operating mode.
PRINT STATE
IF STATE...
Values are:
1 Disconnected
2 Outgoing setup
3 Incoming setup
4 Dial pulses being received
5 Dial pulses being sent
6 Connected

STATUS Displays the state of the call, the call setup mode, the modulo (8 or 128), and type of addressing, and glare condition.
STATUS

SIMP/L COMMANDS

BREAK Disassembles an I–field into its component strings and/or user–defined mnemonics.
BREAK udm,udm,udm
BREAK $A,$B,$C...
BREAK udm,$A,$B,udm,$C...

BUFFER Defines a message for the transmission buffer in hex.
BUFFER = xxx
xxx is the message

BUILD Assembles a message in the transmission buffer.
BUILD udm,udm,udm...
BUILD $A,$B,$C...
BUILD udm,$A,$B,udm,$C...

DEFINE Defines new frame control mnemonic or redefines an existing mnemonic.
DEFINE"name" = x

name is a mnemonic name
x is the field width in bits (maximum width = 16)

LENGTH Returns length of the received frame.
PRINT LENGTH
IF LENGTH...

LRDISPF Outputs the last data field received to a printer or remote device.
LRDISPF

LTDISPF Outputs the last data field built to a printer or remote device.
LTDISPF

RDISPF Displays the last data field received.
RDISPF

REC Transfers the next message in sequence from the reception buffer to the trace buffer.
REC

SLOF Disconnects link by sending a DISC.
SLOF

SLON Attempts to set the frame level link by sending a SABM, SABME, or SNRM.
SLON

STATUS Displays the status of the link.
STATUS

TDISPF Displays the last data field built.
TDISPF

TPRINT Displays the trace buffer.
TPRINT

TRAN Transmits a message.
TRAN

SIMP/L HDLC COMMANDS AND VARIABLES

LNKSTAT Returns the status of the link.
PRINT LNKSTAT
IF LNKSTAT
LNKSTAT values are as follows:

0	Link Disconnected Mode
1	Link Connection Requested
2	Frame Rejected
3	Disconnect Requested State
4	Information Transfer State
5	Local Station Busy
6	Remote Station Busy
7	Local & Remote Stations Busy

SET Sets variable values.

SET N1 = x	Range: 2 – 512
SET N2 = x	Range: 1 – 255
SET T1 = x	Range: 1 – 255
SET WINDOW = x	Range: 1 – 7
SET Network	
SET Subscriber	

SIMP/L SDLC COMMANDS AND VARIABLES

LNKSTAT Returns the status of the link between primary and secondary stations.
PRINT LNKSTAT
IF LNKSTAT
LNKSTAT (primary station) values:

0	Normal Disconnected Mode
1	Link Request State
2	Disconnect Request State
3	Information Transfer State
4	Local Station Busy
5	Remote Station Busy
6	Local & Remote Stations Busy

LNKSTAT (secondary station) values:

0	Normal Disconnected Mode
1	Initialization Mode
2	Frame Reject Mode
3	Information Transfer State
4	Local Station Busy
5	Remote Station Busy
6	Local & Remote Stations Busy

NSI Transmits an NSI frame.
NSI

SET Sets the value of variables and timers.

SET T1=x	Range: 1 – 255
SET T2=x	Range: 1 – 255
SET N2=x	Range: 1 – 99
SET ADDR=x	Range: 0 – FF

TEST Sends a test frame.
TEST

XID Transmits an XID frame.
XID

XIDFLD Sets the data field of an XID frame.
XIDFLD = $A
$A is 6 bytes.

SIMP/L COMMANDS

SIMP/L LAPD COMMANDS AND VARIABLES

* Extendable SIMP/L LAPD only.

***EXTEND** Invokes Extendable SIMP/L LAPD.
EXTEND

FRSTAT Read–only variable. Returns frame status byte of last rec'd data packet.

LNKSTAT Returns the status of the link.
PRINT LNKSTAT
IF LNKSTAT...
LNKSTAT values:

0	Link Disconnected Mode
1	Link Connection Requested
2	Frame Rejected
3	Disconnect Requested State
4	Information Transfer State
5	Local Station Busy
6	Remote Station Busy
7	Local & Remote Stations Busy
8*	Remote Station Not Responding
9	Link Disabled (Multi–Link only)

MOD Sets MOD 8 or MOD 128 sequencing.
MODx x = 8 or 128

SET Sets values of LAPD variables.

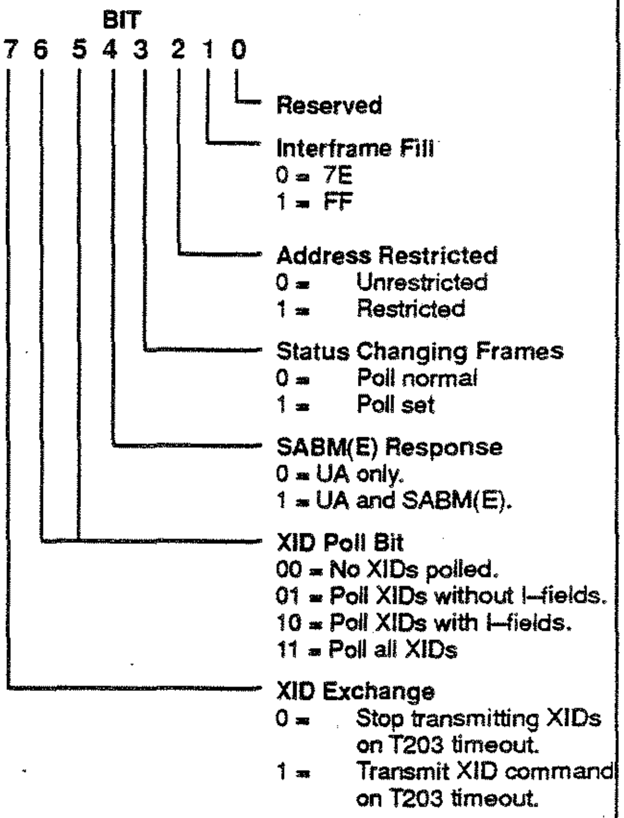
SET N200 = x	Range: 1 – 255
SET N201 = x	Range: 2 – 512
SET SAPI = x	Range: 0 – 63
SET TEI = x	Range: 0 – 127
SET T200 = x	Range: 0 – 255
SET T203 = x	Range: 0 – 255
SET Window = x	Range: 1 – 7
SET Network	
SET Subscriber	

***SET {fnctn}** Sets individual control options in the control configuration byte.
SET fnctn
fnctn is one of the following:

SET	Restrict rec'd responses to transmit SAPI and TEI.
SET UNRESTRICT	Accept responses matching user–defined SAPIs and TEIs and broadcast TEI.
SET SBMCOL	Generate SABM(E) collisions.
SET NOSBMCOL	Stop generating SABM(E) collisions.
SET XIDEXCH	Transmit XID command on T203 timeout.
SET NOXIDEXCH	Stop transmitting XIDs on T203 timeout.
SET POLLSTCH	Set poll bit on status changing frames SABM(E) and DISC.
SET NORMSTCH	Set poll bit normal on status changing frames SABM(E) and DISC.
SET POLALXID	All XIDs polled.
SET ALXIDNPL	All XIDs not polled.
SET POLIXID	Poll XIDs with I–fields.
SET POLNIXID	Poll XIDs without I–fields.

SET FILL=7E	Set interframe fill to 7E.
SET FILL=FF	Set interframe fill to FF.

***SET CONFIG** Sets all control option values by inserting a hex value into the bit–mapped control configuration byte.
SET CONFIG = xx
xx is a hex value.



***SET RSAPI** Sets the values of 1 – 3 user–defined receive SAPIs.
SET RSAPIn = x
n is the SAPI number, range 0 – 2
x is the SAPI value, range 0 – 255

***SET RTEI** Sets the values of 1 – 3 user–defined receive TEIs.
SET RTEIn = x
n is the TEI number, range 0 – 2
x is the TEI value, range 0 – 255

STATUS Displays status information.
STATUS

TRUI Transmits a UI frame
TRUI

***TRXIDC** Transmits an XID command frame.
TRXIDC

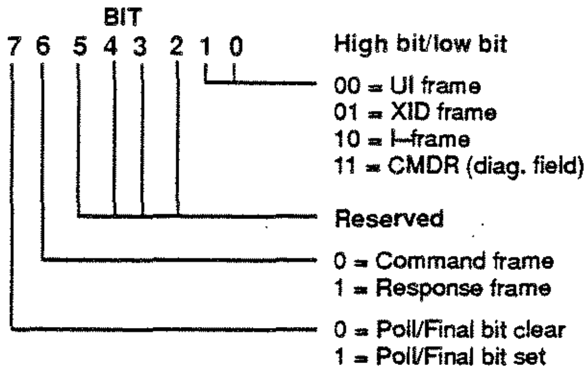
***TRXIDR** Transmits an XID response frame.
TRXIDR

SIMP/L COMMANDS

MULTI–LINK SIMP/L LAPD COMMANDS

FRELNK Read–only variable that returns the number of the lowest disabled link.
PRINT FRELNK
IF FRELNK...
SET LINK = FRELNK
Returns: 0 – 63 Link number
–1 None disabled

FRSTAT Read–only variable that returns a 2–byte frame status value. Interpretation of second byte:



RECLNK Read–only variable that returns the number of the link from which data was last received.
PRINT RECLNK
IF RECLNK...
SET LINK = RECLNK

SET LINK Places one of the 64 available links under user control.
SET LINK = exp exp is in the range 0 – 63

SET SAPI Sets SAPI value for the selected link.
SET SAPI = x x is 0 – 63

SET TEI Sets TEI value for the selected link.
SET TEI = x x is 0 – 127

STATE Displays states of all 64 links.
STATE
Returns: 0 – 9 (see SIMP/L LAPD LNKSTAT for state values)

STATUS Displays status of selected link.
STATUS

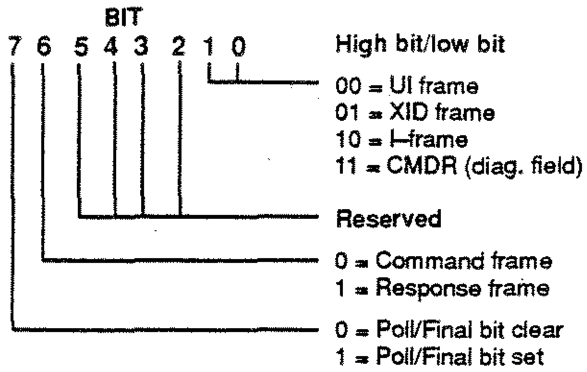
Multi–Link also uses these SIMP/L LAPD commands:

LNKSTAT
SET T200
SET CONFIG
TRXIDR
SET N200
SET T203
TRUI
TPRINT
SET N201
SET Window
TRXIDC

V.120 SIMP/L COMMANDS AND VARIABLES

FRELNK Read–only variable that returns the number of the lowest disabled link.
PRINT FRELNK
IF FRELNK...
SET LINK = FRELNK
Returns: 0 – 63 Link number
–1 None disabled

FRSTAT Read–only variable that returns a 2–byte frame status value. Interpretation of second byte:



RECLNK Read–only variable that returns the number of the link from which data was last received.
PRINT RECLNK
IF RECLNK...
SET LINK = RECLNK

RTRAN Transmits an I–Frame response. The C/R bit is set to 0 in command frames and 1 in response frames.
RTRAN

SET LINK Places one of the 64 available links under user control.
SET LINK = exp exp is in the range 0 – 63

SET LLI Sets the LLI value of the selected link.
SET LLI = x x is 0 – 8191

STATE Displays the states of all 64 links.
STATE
Returns: 0 – 9 (see SIMP/L LAPD LNKSTAT for state values)

STATUS Displays status of selected link.
STATUS

V.120 SIMP/L also uses these SIMP/L LAPD commands:

LNKSTAT
SET T200
SET CONFIG
TRXIDR
SET N200
SET T203
TRUI
TPRINT
SET N201
SET Window
TRXIDC

ASYNC BASIC COMMANDS

BREAK Transmits a BREAK sequence.
BREAK

CRC16 Calculates the CRC for a string
$B=CRC16($A)

FOXMESS Transmits the standard FOX message and repeats it until the operator hits any key.
FOXMESS

FRAMING Returns a value that indicates the presence of stop bits at the end of the received block.
PRINT FRAMING
IF FRAMING...
FRAMING returns:
FRAMING=0 Stop bits
FRAMING=1 No stop bits

LENGTH Returns the number of characters received in a block.
PRINT LENGTH
IF LENGTH...

LRC Calculates the LRC for a string.
$X = LRC($Y)

PARITY Indicates whether a parity error has occurred.
PRINT PARITY
IF PARITY...
PARITY returns:
PARITY =0=No parity error
PARITY =1=Parity error

REC Assigns the next character (if in character mode) or block of characters (if in block mode) to string variables.
REC $A, $B, $C...

RXBREAK Indicates if a break sequence has been received.
PRINT BREAK
IF BREAK...
BREAK returns:
RXBREAK=0=No break sequence
RXBREAK=1=Break sequence

TPRINT Displays the contents of the trace buffer.
TPRINT Prints the trace in ASCII.
TPRINT HEX Prints the trace in hexadecimal.

TRAN Transmits data in strings, mnemonics or literal data.
TRAN $A...
TRAN CR, LF...
TRAN "ABCD"...
TRAN $A, CR, LF, "ABCD"

BSC BASIC COMMANDS

CRC16 Calculates the two–byte CRC for a string.

```
CRC16($A)
```

IDLE Determines what is transmitted when the line is idle.

```
IDLE=SYNC
IDLE=MARK
```

LRC Calculates the LRC for a string.

```
LRC($A)
```

REC Takes the next received block from the acquisition buffer.

```
REC
```

RXLENGTH Returns the length of the last received block.

```
PRINT RXLENGTH
IF RXLENGTH...
```

TPRINT Displays the contents of the trace buffer.

```
TPRINT            Displays the trace buffer in hex.
TPRINT ASCII      Displays the trace buffer in ASCII.
TPRINT EBCDIC     Displays the trace buffer in EBCDIC.
```

TRAN Transmits a block from the transmission buffer (TXBUFFER), according to the framing defined by the transmission control status byte (TXSTATUS).

```
TRAN
```

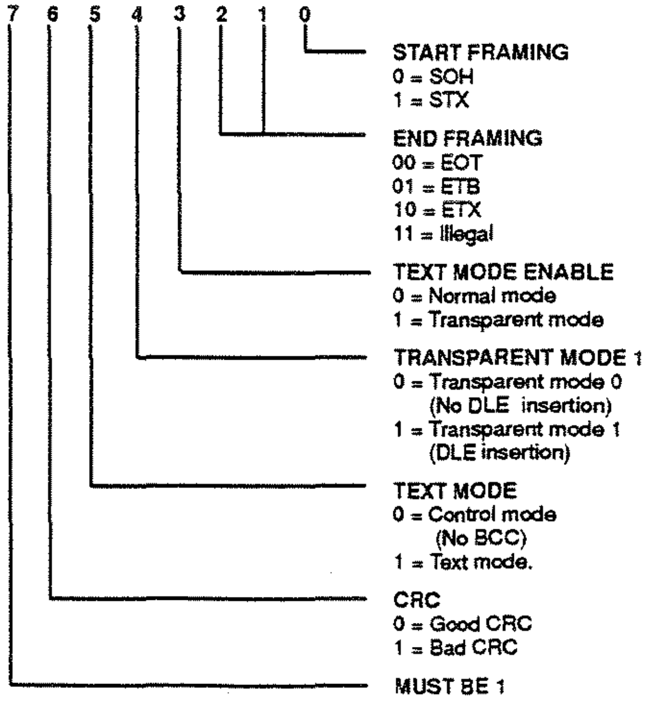
TXBUFFER Defines the contents of the transmission buffer

```
TXBUFFER = DLE
TXBUFFER = ACK
TXBUFFER = 0
TXBUFFER = $A
TXBUFFER = &10, &70
TXBUFFER = DLE, $A, &70
```

TXSTATUS Defines the transmission control status byte, as shown below.

```
TXSTATUS = &xx
TRAN $A, CR, LF, "ABCD"
```

xx is a hex value



START FRAMING
0 = SOH
1 = STX

END FRAMING
00 = EOT
01 = ETB
10 = ETX
11 = Illegal

TEXT MODE ENABLE
0 = Normal mode
1 = Transparent mode

TRANSPARENT MODE 1
0 = Transparent mode 0
(No DLE insertion)
1 = Transparent mode 1
(DLE insertion)

TEXT MODE
0 = Control mode
(No BCC)
1 = Text mode.

CRC
0 = Good CRC
1 = Bad CRC

MUST BE 1

SITREX COMMANDS

FRAME LEVEL COMMANDS

Send User–Defined Frame

FBb........b	Frame defined in binary.
FAa........a	Frame defined in ASCII.
FHh........h	Frame defined in hex.

Send Unnumbered Commands

F(P)DISC	Polled/unpolled DISC on primary address.
F(P)SABM	Polled/unpolled SABM on primary address.

Send Unnumbered Responses

F(F)UA	UA frame.
F(F)DM	DM frame.
F(F)CMDRh1h2(Vs)(,Vr)(B)(W)(X)(Y)(Z)	CMDR frame.

Send Numbered Commands

FPRR(Nr)	Sends an RR.
FPRNR(Nr)	Sends an RNR.

Nr is in the range 0 –7.

Numbered Responses

F(F)RNR(Nr)	Sends a RNR frame.
F(F)RR(Nr)	Sends a RR frame.
F(F)REJ(Nr)	Sends a Rej frame.

Nr is in the range 0 –7

Send Information Frame

F(P)I(Ns)(,Nr)(PACKET)	Packet mnemonic.
F(P)I(Ns)(,Nr)(PHh1h2...)	Packet in hex.
F(P)I(Ns)(,Nr)(PAabcd...)	Packet in ASCII.

PACKET LEVEL COMMANDS

Send User–Defined Packet

PHh........h	Packet in hex.
PAa........a	Packet in ASCII.

Send Call/Call Confirmation Packets

PUnCALL(D)(Na or V)(,Nb)(,Fh...h)(,DHh...h)
PUnCALL(D)(Na or V)(,Nb)(,Fh...h)(,DAa...a)
Sends a Call packet.
PUnCCALL(D)
Sends a Call Confirmation packet.
n = pseudo–user, range 1 – 7.
D = delivery confirmation bit.
Na = called address in decimal
V = called number configured in SITREX menu
Nb = calling number in decimal
F = called facilities
DA = Data in ASCII
DH = Data in hex

Send Supervisory Packets

PUnRR(Pr)	Sends an RR packet.
PUnRNR(Pr)	Sends an RNR packet.
PUnREJ(Pr)	Sends a Reject packet.

n = pseudo–user, range 1 – 7.
Pr, range 0–7 (Mod 8); range 0–127 (Mod 128)

PACKET LEVEL COMMANDS (cont.)

Send Restart/Restart Confirmation Packets

PRST(h1h2)(h3h4)   Sends a Restart packet
h1h2 = Cause code
h3h4 = Diagnostic code
PCRST   Restart Confirmation packet.

Send Clear/Reset/Interrupt Confirmation

PUnCLEAR(h1h2)(h3h4)(,DHh...h)
Clear packet with data in hex.
PUnCLEAR(h1h2)(h3h4)(,DAa...a)
Clear packet with data in ASCII.
PUnCCLEAR
Clear Confirmation packet.

PUnRSET (h1h2)(h3h4)	Reset packet.
PUnCRSET	Reset Confirmation.
PUnINT(h1h2)(h3h4)	Interrupt packet.
PUnCINT	Interrupt Confirmation.

DA = Data in ASCII
DH = Data in hex
h1h2 = cause code
h3h4 = diagnostic code

Send Data Packets

PUnD(Ps)(Pr)(Q)(M)(D)Hh....h	Hex.
PUnD(Ps)(Pr)(Q)(M)(D)Aa....a	ASCII.

n = pseudo–user, range 1 – 7.
Q = Qualifier bit.
M = More Data bit.
D = Delivery confirmation bit.

Send Diagnostic Packet

Sends a diagnostic packet on logical channel 0.
PDIAGh1h2h3h4h5h6h7h8
h1h2= diagnostic byte
h3 – h8 = first 3 bytes of header information

DISPLAY AND PRINT COMMANDS

Display User Parameters

DISPT	Displays timer values in decimal.
LDISPT	Prints the timer values in decimal.
DISPC	Displays counters in hex.
LDISPC	Prints counters in hex.
DISPV	Displays variable values.
LDISPV	Prints variable values
DISPX	Displays numeric variables in hex.
LDISPX	Prints numeric variables in hex.
DISPM	Displays length (decimal) and contents of message buffer (hex).
LDISPM	Prints length (decimal) and contents of message buffer (hex).

Print

PRINT *text*	Displays *text* on the screen.
LPRINT *text*	Outputs *text* to a printer.

List Scenario

LIST	Displays the scenario in memory.
LLIST	Prints the scenario in memory.

SITREX COMMANDS

PARAMETER COMMANDS

Set Frame Level

SFON	Sets frame level ON.
SFOF	Sets frame level OFF.

Set Packet Level

SPON	Set packet level ON.
SPOF	Set packet level OFF.

Set Link Level

SLON	Sets Link ON.
SLOF	Sets Link OFF.

Force Link On

LNKUP

Forces the Simulator to assume that the link has already been established.

Transmit CRC

SCRC+	Frames include good CRC.
SCRC–	Frames sent without CRC.

Set Frame and Packet Window Size

SKx

x = frame window size, range 1 – 7

SUnW(R or T)x

x = packet window size, range 1 – 7

T = Transmit window size.

R = Receive window size.

n = pseudo–user number,range 1 – 7

Set Frame and Packet State Variables

SNS	Increments N(s).
SNS–	Decrements N(s).
SNSx	Sets N(s), range 0 – 7
SNR+	Increments N(r).
SNR–	Decrements N(r).
SNRx	Sets N()r, range 0 – 7
SUnPR+	Increments P(r).
SUnPR–	Decrements P(r).
SUnPRx(xx)	Sets P(r), range 0 – 7 (Mod 8); range 000 – 127 (Mod 128)
SUnPS+	Increments P(s).
SUnPS–	Decrements P(s).
SUnPSx(xx)	Sets P(s), range 0 – 7 (Mod 8); range 000 – 127 (Mod 128)

Set Data Packet Length

SUnLTnnn	Sets maximum length of transmitted data packet.
SUnLRnnn	Sets maximum length of received data packet.
SGTh1h2	Sets the length of the data in the data packet sent by a traffic generator

PARAMETER COMMANDS (CONT.)

Set Primary/Secondary Address

SPAh1h2	Sets Primary Address.
SSAh1h2	Sets Secondary Address.

h1h2 is the address.

Set Logical Channel Group Number

SLGh1

h1 = LCGN in hex

Assigns a default LCGNfor the next placed call.

Set Interface Leads

Snnn+	Sets interface lead nnn active (space).
Snnn–	Sets interface lead nnn inactive (mark).

nnn is one of the signal numbers:

DCE	106	CTS
	107	DSR
	109	DCD
	122	SDCD
	125	RI
DTE	105	RTS
	108	DTR

Test Interface Leads

Tests interface lead for mark or space and jumps to line number dddd if test is true.

IFnnn+dddd	Interface signal is active.
IFnnn– dddd	Interface signal inactive.

Set Timers

ST'h1h2h3h4	Sets timer T'
ST"h1h2	Sets timer T"
STUh1h2h3h4	Sets user–defined timer TU.

Set Counters

SCnh1h2	Sets counter n to hex value h1h2.
SCn+	Increments counter n.
SCn–	Decrements counter n.

n is in the range 0 – 7.

Set Pseudo–User Type

Defines pseudo–users 3 – 7.

Pseudo–user 1 is reserved for the trace page. Pseudo–user 2 is reserved for the Simulation page.

SPUnA	Pseudo–user is a Data Absorber.
SPUnE	Pseudo–user is an Echo Generator.
SPUnT	Pseudo–user as a Traffic Generator

Set Up PVCS and SVCS

SUnVPh1h2h3	Sets up a PVC.
SUnVCh1h2h3	Sets up an SVC.

n = pseudo–user number

h1 = LCGN

h2h3 = LCN

SITREX COMMANDS

NUMERIC VARIABLE COMMANDS

Variable Operations

SXAHh1h2	Assigns a value to XA, in hex.
SXA+XB	Adds XA and XB and stores result in XA.
SXA–XB	Subtracts XB from XA and stores result in XA.
SXA.XB	Logical AND.
SXA/XB	Logical OR.
SXA@XB	Logical Exclusive OR (XOR).

Shift and Rotate

SXADn	Shifts XA n times to the right.
SXAGn	Shifts XA n times to the left.
SXARn	Rotates XA n times to the right.
SXALn	Rotates n times to the left.

n is in the range 1 to 7.

Keyboard Input

Scenario waits for the user to enter a two–digit hex value and then assigns the value to a numeric variable

INPUT XA

Test Variables

Tests variable using relational operators as shown below. If true, scenario jumps to line dddd.

IXA=XB dddd	XA equals XB
IXA=h1h2 dddd	XA equals h1h2.
IXA#XB dddd	XA is not equal to XB.
IXA#h1h2 dddd	XA is not equal to h1h2.
IXA<XB dddd	XA is less than XB.
IXA<h1h2 dddd	XA is less than h1h2.
IXA>XB dddd	XA is greater than XB.
IXA>h1h2 dddd	XA is greater than h1h2.
IXA(XB dddd	XA <= to XB.
IXA(h1h2 dddd	XA <=to h1h2.
IXA)XB dddd	XA >= XB.
IXA)h1h2 dddd	XA >= h1h2.

MESSAGE BUFFER COMMANDS

Assign Contents of Message Buffer

SMHh...h	Writes hex values into buffer, where h...h is up to of 128 hex digits.
SMAa...a	Writes ASCII characters into buffer, where a...a is a maximum of 64 ASCII bytes.
SXAIXB	Inserts value of variable XA in the byte of the buffer indicated by the value contained in variable XB.
SXAIh1h2	Inserts the value of XA in the byte of the buffer indicated by the 2–digit hex value h1h2.

Message Buffer Length

SXAI00	Message buffer length = XA.
SXAO00	Extracts buffer length and stores it in XA.

Byte Extraction

SXAOXB	Extracts byte at location indicated by XB, and stores it in XA.
SXAOh1h2	Extracts byte at location indicated by the hex value h1h2, and stores it in XA.

Test Message Buffer Contents

Compares the contents of the message buffer to the byte and mask configuration in the command.

ISXX/YY(,XX/YY)(....) dddd

dddd is the line number

Transmit Message

FM	Transmits the frame, assigning the first byte of the message buffer (byte 1) to the first byte of the frame.
PM	Assigns the contents of the message buffer, excluding the message buffer length (byte 0), to the I–Field of a frame (byte 3 and following) and transmits it.
PUnDS	Assigns the contents of the message buffer (beginning with byte 1) to the data portion of the I–Field, and transmits it from the pseudo–user n.

TRACE BUFFER COMMANDS

Display Trace

TPRINT	All levels interpreted, data in hex.
TPRINTA	All levels interpreted, data in ASCII.
TPRINTF	All levels uninterpreted, in hex.
TPRINTP	Interpret frame–level, I–field in hex.

Load Trace File

TLOAD

Print Trace

LTPRINT	All levels interpreted, data in hex.
LTPRINTA	All levels interpreted, data in ASCII.
LTPRINTF	All levels uninterpreted, in hex.
LTPRINTP	Interpret frame–level, I–field in hex.

Save Trace

TSAVE"0*filename*"	Save to the hard disk.
TSAVE"1*filename*"	Save to floppy disk.

Set Trace On/Off

STON	Sets the trace buffer ON.
STOF	Sets trace buffer OFF.

Clear Trace

TRACE

Trace Length

Defines number of data bytes (0 – 255) displayed by the trace buffer.

STRh1h2

h1h2 is a hex value in the range 0 to FF.

SITREX COMMANDS

WAIT COMMANDS

These commands wait for the specified item before continuing the scenario.

WF(command)	Waits for frame type.
WP(command)	Waits for packet type.
WTXX/YY(,XX/YY)(...)	Waits for byte mask.

Wait and Store Commands

These commands wait for the specified item and then store it in the message buffer.

WSF(command)	Waits for frame type.
WSP(command)	Waits for packet type.
WSTXX/YY(,XX/YY)(...)	Waits for byte mask.

Wait Watchdog Timer

Sets the Watch Dog Timer for WAIT commands.
SWTxxxx
xxx = tens of milliseconds (.0001)

Wait Jump Addresses

Watchdog Address

Sets jump address for the Watch–Dog Timer.
SADRWT dddd
dddd=line number

Wait Jump Address

Jumps to line number if the received item is not the one specified in the WAIT command.
SELSE dddd
dddd= line number

PROGRAM MANAGEMENT COMMANDS

Chain Program

Loads and executes a scenario.
&xfilename
x is 0 (hard drive) or 1 (floppy drive)

Load Program

Loads a scenario file into memory.
LOAD
When prompted for a filename, use format:
xfilename
x is 0 (hard drive) or 1 (floppy drive)

Remark

Enables you to enter programming remarks in a scenario.
REM *(text)*

New Program

Erases the scenario in memory so that a new program can be written.
NEW

Run Program

Executes the scenario in memory.
RUN

Save Program

Saves the scenario in memory to disk.
SAVE
When prompted for a filename, use format:
xfilename
x is 0 (hard drive) or 1 (floppy drive)

MISCELLANEOUS COMMANDS

Set Up Program Loop

*h1h2	Beginning of loop
;	End of loop.

h1h2=times to execute loop, range 1 – FF

Conditional Jump (IF)

Tests a timer or counter for 0. If test is true, jumps to line dddd. Otherwise, the next command will be executed.

IFT' dddd	Tests timer T'.
IFT" dddd	Test timer T".
IFTU dddd	Tests user–defined timer TU.
IFCn dddd	Tests counter n (range 0 – 7)

Unconditional Jump (GOTO, GOSUB)

GOTO dddd	Jump to line number dddd.
GOSUB dddd	Jump to line number dddd and execute command until RETURN is encountered.
RETURN	SEnd of GOSUB subroutine.

Reinitialization

HALT	Exits SITREX and returns to the Chameleon 32 main menu.
EXIT	Exits SITREX command mode and returns to the SITREX Automatic X.25 Simulator.
ESCape key	Stops scenario execution.

SITREX AUTOMATIC SIMULATOR COMMANDS

COMMAND	FUNCTION
P1	Sets the Automatic X.25 Simulator to echo Called and Calling Addresses in Call Confirmation packets.
PP	Activates the Trace Buffer so that traffic is stored and displayed in the trace page. Once the trace is active, use CTRL P to make the trace idle.
PS	Enters programming mode and displays the ! prompt enabling you to enter Sitrex commands and write programs. From the ! prompt, you can exit Sitrex using the HALT command or exit program mode using the EXIT command.

SITREX DEFAULT PSEUDO–USERS

NUMBER	CONFIGURATION
01	Reserved for the Chameleon 32 Trace page (pink window).
02	Reserved for the Chameleon 32 Simulation page (blue window).
03	Traffic Generator
04	Echo Generator
05	Data Absorber
06	Second Traffic Generator
07	Third Traffic Generator

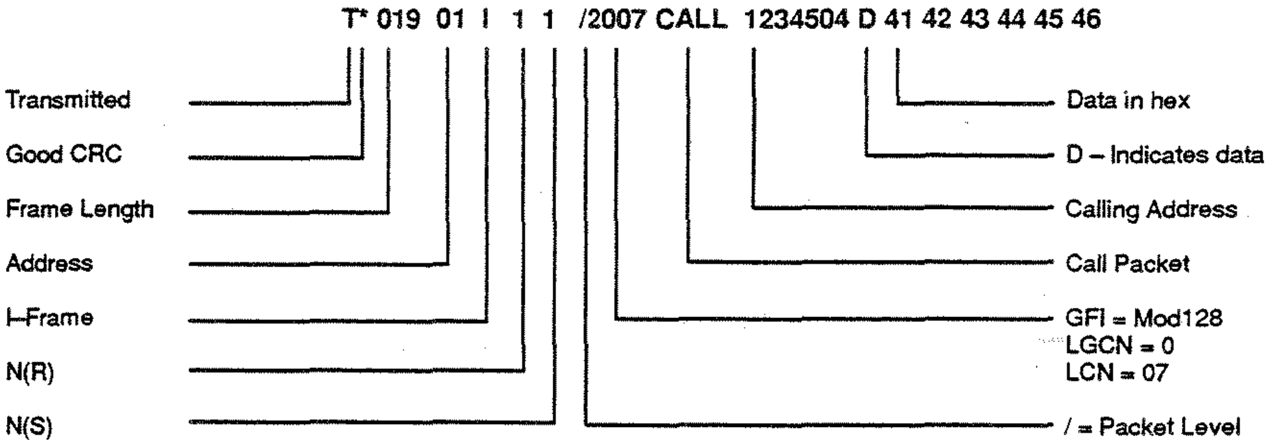
SITREX TRACE PAGE COMMANDS

The table below lists the commands that control the display of traffic in the SITREX trace page.

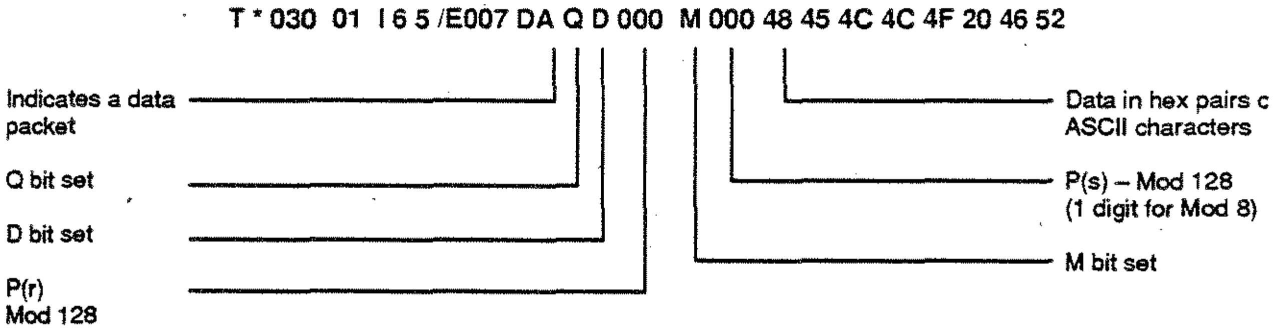
COMMAND	FUNCTION
0–9	Modifies the scrolling speed. Fastest = 1, Stop = 0
A	Toggles between ASCII and hex as format of displayed data.
F	Toggles display of frame level interpretation on/off.
P	Toggles display of packet level interpretation on/off.
R	Re–displays the contents of the trace.
CTRL C	Clears the trace memory.
CTRL P	Exits the trace mode and returns to the base Simulator level.

SITREX TRACE INTERPRETATION

The example below show the interpretation of a SITREX trace display. The first example interprets a display of a transmitted CALL packet. The second example interprets a transmitted DATA packet.



```
T* 019 01 I 1 1 /2007 CALL 1234504 D 41 42 43 44 45 46
```



```
T * 030 01 I 6 5 /E007 DA Q D 000 M 000 48 45 4C 4C 4F 20 46 52
```

Indicates a data packet
Q bit set
D bit set
P(r) Mod 128
Data in hex pairs c ASCII characters
P(s) – Mod 128 (1 digit for Mod 8)
M bit set

SITREX ERROR CODES

ERROR	MEANING
00	First character incorrect.
04	Illegal Line number (valid range is 0 to 9999).
15	Attempt to re–assign a new LCN to a previously set pseudo–user.
16	Attempt to give a previously assigned LCN to a new pseudo user.
18	RETURN without GOSUB.
20	Incomplete Loop (; before * nn).
21	Line number specified does not exist.
22	Attempt to send a data packet on a logical channel that i not set up.
26	Error in the call facility field of a call packet.
81–83	No space for scenarios (memory full).

The message P followed by a two–digit code is Packet Error Coding from the Automatic X. 25 Simulator.

P00	Restart at the packet level.
P01	Internal error.
P02	Flow control anomaly (bad P(s) or P(r)).
P03	Call or interrupt collision on a logical channel.

TABLE ERROR 01,02	Internal error.
TABLE ERROR 06,07	Internal error.
TABLE ERROR 0A,0B	Internal error.
TABLE ERROR 16	This error is associated with high density traffic.

T00	Reception of a frame with an I–field that exceeds N1.
T01	Address unknown, frame ignored (Not Primary or Secondary address).
T02	Response with poll final bit set when not solicited.
T03	Response with poll final bit not set when solicited.
T04	Response unknown, results in sending CMDR.
T05	Incorrect length of response frame, results in CMDR.
T08	Received unsolicited UA.
T09	Incorrect Nr received.
T10	Reject frame received.
T11	RNR frame received.
T14	Unknown command received.
T15	Incorrect Ns received.
T16	I–Frame out of sequence and station not busy and no Tx.RNR requested and no prior Tx.REJ OR Tx.P/F set.
T17	I–frame out of sequence and station not busy and no Tx.RNR requested and no prior Tx.REJ and no Tx.P/F set.
T18	Received frame out of sequence with Tx.RR request.
T19	Received frame out of sequence.
T20	Internal error.
T21	Error which occurs when SITREX sends a disconnect and the device under test gives an unexpected response.
T22–24	Internal error.

C SHELL COMMANDS

& Runs program in background mode.
progname &

Programmer's remark.
#text

' Echoes text to screen.
'text

batch Executes a batch file
batch *filename*

cat Concatenates and prints files
cat *filename*

cd Changes current directory
cd *path*

cp Copies files into a directory
cp *oldfile newfile*

ctags Creates a file named *tags* that references the functions in the target C program files. The *tags* file can then be used to locate functions while using the vi editor.
ctags files

dump Prints files in hex to standard output.
dump *filename filename...*

exit Exits C shell. Returns to main menu.
exit

format Formats a floppy disk.
format

getenv Displays environmental variable
getenv *name*

Name:	BC	Background color
	FC	Foreground color
	HOME	Default cd path
	PATH	Search path
	YEAR	global–curr–year
	(or user–defined variable)	

help Lists built–in shell commands
help

jobs Prints job control status
jobs

kill Kills a process that is running
kill *pid* (pid=process ID)

ls Lists files
ls [–d] [–k] [–l] [–s] [spec]

–d	Sorted by date
–k	Sorted by file extension
–l	Long list format
–s	Sorted by size
spec	Filename or path specification

man Displays the named help file.
man *filename*

mkdir Creates a subdirectory
mkdir *dirname*

mkres Makes a program RAM resident
mkres [–p] prog

–p Cannot be removed by memory manager

more Displays file or pipe output, one screen at a time
more *filename*

mv Moves a file

mv file1 file2	Replace file2 with file1
mv file dir	Move file to directory

pwd Prints current subdirectory
pwd

rm Deletes one or more files
rm *filename filename...*

rmdir Deletes a subdirectory
rmdir *dirname*

rmres Removes a program from residency
rmres *pid* (pid=process ID)

run Runs a program as a separate process.
run[–xxx] <prog> &

–xxx Priority in the range 1 – 230 (230 = highest priority)
progprogram filename

setenv Sets an environment variable
setenv *name 'value'*

name:	BC	Background color
	FC	Foreground color
	HOME	Default cd path
	PATH	Search path
	YEAR	global–curr–year
	(or user–defined variable)	

shell Opens a new page with the C shell.
shell &

size Prints file size to standard output
size *filename filename...*

time Displays the system time.
time

C SHELL REFERENCE

COMPILER COMMANDS

cc Compiles and links files

cc [–c] [flags] [file.c/file/o...]

–c	Compiles only, does not link
flags	Flags for ld and mcc
files	C Source or Object file

mcc Compiles C source files

mcc[–dname[=value]][–Ipath] [–x] cfile

–Dname	Same as #define name in source
–Dname=value	Same as #define name value in source
–Ipath	Searches path for Include file.
–x	Trace mode.
cfile	C source file name

LINKER COMMAND

ld Combines object files into one executable program

ld [–V] [–Llib] [–M] [–X] [–Txxx] [–o output] <objects> [libraries]

–V	Verbose option.
–Llib	Library search path.
–M	Prints names and addresses of globals.
–X	Debug option.
–Txxx	Causes the linker to adjust references within the program as if the program was at hex memory location xxx.
–o output	Output file.
objects	Input object files. This must always include: /lib/init.o
libraries	One or more input library files, if not already specified with the –Llib option.

LIBRARIAN COMMAND

ar Groups files into a single archive (object file libraries)

ar key [pos] afile file

key	One of the following commands:
	t – Table of contents
	r – Replace/add file
	ra – Replace after [pos]
	d – Delete file
	x – Extract file
	w–Write file to stdout
	v – verbose
	l – Create random library
pos	Position in archive to add file
afile	Archive file name
file	Filename according to key

MAKE UTILITY

make Executes commands in a makefile, causing related program files to be updated

make[opt] [–f mfile] file

opt	One of the following options:	
	–i	Ignore error codes
	–k	Abandon work on current entry
	–n	No execute mode
	–r	Do not use built–in rules
	–s	Silent mode.
–f mfile		Name of makefile to execute
file		Names of file(s) to update

DISASSEMBLER COMMAND

dis Allows you to check the compiler code generation

dis [–n] [–r] [–a] [–i] ofile

–n	No symbol name conversion
–r	Print Bcc instructions
–a	Print as an assembly file
–i	Print hex value of instruction
ofile	Object filename

EGREP COMMAND

egrep Searches files for user–defined patterns.

egrep.ttp [–C][–L][–V][–N][–S] pat[files]

Options:

–C	Prints number of lines matching pattern.
–L	Prints file names matching pattern.
–V	Prints line that do *not* match pattern.
–N	Prints the line number of the line matching pattern
–S	Silent. Prints only error messages.

pat User defined pattern to match.

\x	Matches character *x*.
^	Matches beginning of line.
.	Matches any character.
[abc]	Matches characters a, b, c.
[a–z]	Matches characters in the range a to z.
*	Zero or more matches of the regular expression preceding *
+	One or more matches of the regular expression preceding+
?	Zero or one matches of the regular expression preceding ?
|	Two regular expressions separated by | match either a match for the first or a match for the second.

C LIBRARY FUNCTIONS

C LIBRARY FILENAME: libc.a

abs
Returns absolute value.
# include <stdio.h>
int abs (i)
int i;

alloca
Allocates RAM on the stack.
char *alloca(size)
unsigned int size;

atof
Converts ASCII string to a floating-point number.
double atof (nptr)
char *nptr;

atoi
int atoi(str)
char *str;

atol
long atol(str)
char *str;

bcmp
Compares two blocks of memory.
int bcmp(block1, block2, len)
char *block1, *block2;
int len;
Returns: 1 = blocks are identical

bcopy
Copies a block of memory to another block.
int bcopy(source, destin, len)
char *source, *destin;
int len;

bzero
Zeroes a block of memory.
int bzero(block1, len)
char *block1;
int len;

calloc
Allocates RAM for array of *nelem* elements of *elsize* size.
char *calloc(nelem,elsize)
unsigned int nelem,elsize;

clearerr
Resets error and EOF indicators to 0.
#include <stdio.h>
clearerr (stream)
FILE *stream;

close
Closes a file.
int close (fildes)
int fildes;
Returns: 0 = Successful
−1 = Error

creat
Create file or overwrite existing file.
int creat(fname, oflag)
char *fname;
int oflag;
Returns: 0 = Successful
−1 = Error

execl
Executes a file.
execl(name, arg0, arg1,...,argn,OL)
char *name;
char *arg0,arg1,...,argn;

execv
Executes a file.
execv(name,argv)
char *name, *arvg[];

exit/_exit
Terminates a process. _exit returns without performing cleanup operations.
exit (status)
_exit (status)
int status;

fclose
Writes all buffered data and closes stream.
#include <stdio.h>
int fclose(stream)
FILE *stream;

feof
Indicates when end–of–file is detected when reading stream.
#include <stdio.h>
int feof(stream)
FILE *stream;
Returns: 0 =Not EOF
Non–zero = EOF detected

ferror
Indicates when an I/O error occurs reading to/writing from the stream.
#include <stdio.h>
int ferror(stream)
FILE *stream;
Returns: 0 =No error
Non–zero = Error detected

fflush
Writes buffered data, but stream remains open.
#include <stdio.h>
int fflush(stream)
FILE *stream;
Returns: 0 = Successful
EOF = Error

fgetc
Same as getc, but is a true function.
include <sdtio.h)
int fgetc(stream)
FILE *stream;

fgets
Reads characters from stdin into array pointed to by *s* until EOF, new line, n–1 characters are read.
#include <sdtio.h>
char *fgets(s,n,stream)
char *s;
int n;
FILE *stream;

C LIBRARY FUNCTIONS (CONTINUED)

fileno
Returns the integer file descriptor.
#include <stdio.h>
int fileno(stream)
FILE *stream;

fopen
Opens a file and associates a stream with it.
#include <stdio.h>
FILE *fopen(file_name,type)
char *file_name;
char *type; w = Create for writing
r = Open for reading
a = Append
r+ = Open for update
w+ = Create for update
a+ = Random open

fprintf
_fprintf
Places output to a named output stream.
#include<stdio.h>
int fprintf(stream,format[,arg]...)
FILE *stream;
char *format;

fputc
Similar to putc, but is a true function.
#include <stdio.h>
int fputc(c,stream)
char c;
FILE *stream;

fputs
Writes the string pointed to by *s* to *stream.*
#include<stdio.h>
int fputs (s,stream)
char *s;
FILE *stream;

fread
Reads *nitems* of data from input stream at *ptr* and places in array.
#include <stdio.h>
int fread (ptr,size,nitems,stream)
char *ptr;
int size, nitems;
FILE *stream;

free
Makes RAM at *ptr* available for allocation.
free(ptr)
char *ptr;

freopen
Substitues a file in place of the open stream.
#include <stdio.h>
FILE *freopen (file_name,type, stream)
char *file_name, *type;
FILE *stream;
char *type; w = Create for writing
r = Open for reading
a = Append
r+ = Open for update
w+ = Create for update
a+ = Random open

fscanf
Reads from named input stream and converts formatted input.
#include <stdio.h>
int fscanf(stream,format [,pointer]..)
FILE *stream;
char *format;

fseek
Sets position of next I/O operation on the stream.
#include<stdio.h>
int fseek(stream, offset,ptrname)
FILE *stream;
long offset; Range 0 – 2
int ptrname;
Returns: 0 = Successful
Non–zero = Error

ftell
Returns offset of current byte relative to beginning of file.
long ftell(stream)
FILE *stream;

fwrite
Appends *nitems* of data from array at *ptr* to an output stream.
#include <stdio.h>
int fwrite(ptr,size,nitems,stream)
char *ptr;
int size,nitems;
FILE *stream;

getc
Returns the next byte in *stream* and advances pointer one byte.
#include <stdio.h>
int getc(stream)
FILE *stream;

getchar
Returns the next character from stdin.
#include <stdio.h>
int getchar()

gets
Reads characters from stdin into array pointed to by *s* until EOF or new–line.
#include <stdio.h>
char *gets(s)
char *s;

getw
Returns next word or integer from input stream.
#include<stdio.h>
int getw(stream)
FILE *stream;

is...
Character–coded integer values.
#include <ctype.h>

int isalpha c	Letter
int isupperc	Upper case letter
int islowerc	Lower case letter
int isdigit c	Digit
int isalnum c	Alphanumeric
int isspace c	Space, tab, CR, newline, form feed
int ispunct c	Punctuation
int isprint c	Printi chars: 040–0176
int iscntrl c control	Delete char. 0177 or chars < 040
int isascii c	ASCII chars: < 0200
int isxdigit	Hex digit

int c;
Returns: 0 = False
Non–zero=True

C LIBRARY FUNCTIONS (CONTINUED)

lmalloc Returns pointer to a block of RAM of *size* or greater. Like malloc but accepts a long parameter.

```
char *lmalloc(size)
long size;
```

longjmp Restores the environment saved by setjmp in env.

```
#include<stdio.h>
longjmp(env,val)
jmp–buf env;
int val;
```

lrealloc RAM allocator which changes the size of the block pointed to by *ptr* to *size* bytes. Like realloc, but accepts a long parameter.

```
char *lrealloc(ptr, size)
char *ptr;
unsigned long size;
```

lseek Moves the file pointer according to *whence*.

```
long lseek(fildes,offset,whence)
int fildes;
long offset;
```

int whence; 0 = Set to *offset*
1 = Curr. position+*offset*
2 = File size + *offset*

malloc Returns pointer to a block of RAM (64K bytes or less) of *size* or greater.

```
char *malloc(size)
unsigned int size;
```

open Opens a file described for fname and sets flag to oflag.

```
#include <fnctl.h>
int open(fname,oflag)
char *fname;
```

int oflag; O–RDONLY Read only
O–WRONLY Write only
O–RDWR Read/write
O–BINARY Binary mode

perror Writes an error mesage onto the standard stream.

```
perror(s)
char *s;
extern int sys_nerr;
extern char *sys_errlist[];
```

printf Places output on stdout.

```
#include <stdio.h>
int printf(format[,arg]...)
char *format;
```

putc Writes character *c* to the output stream at current pointer position.

```
#include<stdio.h>
int putc (c,stream)
char c;
FILE *stream;
```

putchar Equivalent to putc(c,stdout).

```
#include<stdio.h>
int putchar(c)
char c;
```

puts Writes the string pointed to by *s* to stdout.

```
#include <stdio.h>
int puts(s)
char *s;
```

putw Writes the word (integer) *w* to the output stream at current pointer position.

```
#include <stdio.h>
int putw (w,stream)
int w;
FILE *stream;
```

qsort Quick sort algorithm.

```
qsort(base, nelem, width, compare)
char *base;
int nelem, width;
int (*compare) ();
```

rand Generates a random number.

```
#include <stdio.h>
int rand()
```

read Reads *nbyte* bytes from the file into the buffer pointed to by *buf*.

```
int read(fildes,buf,nbyte)
int fildes;
char *buf;
unsigned int nbyte;
```

realloc RAM allocator which changes the size of the block pointed to by *ptr* to *size* bytes.

```
char *realloc(ptr, size)
char *ptr;
unsigned int size;
```

rename Renames a file on disk.

```
int rename (from, to)
char *from, *to;
```

rewind Equivalent to fseek(stream,OK,0), but no value is returned.

```
#include <stdio.h>
rewind(stream)
FILE *stream;
```

scanf Reads from stdin and converts formatted input.

```
#include <stdio.h>
int scanf(format[,pointer]...)
char *format;
```

setbuf Assigns buffer pointed to by *a* to a stream.

```
#include <stdio.h>
setbuf(stream,buf)
FILE *stream;
char *buf;
```

setjmp Non–local goto which saves its stack environment in *env* for use by longjmp.

```
#include <stdio.h>
int setjmp(env)
jmp–buf env;
```

C LIBRARY FUNCTIONS (CONTINUED)

sprintf
_sprintf Places output in consecutive bytes starting at *s*.
#include<stdio.h>
int sprintf (s,format[,arg]...)
char *s, format;

srand Resets the random number generator to a new starting point.
#include <stdio.h>
srand(seed)
long seed;

sscanf Reads from character string a and converts formatted input.
#include <stdio.h>
int sscanf(s,format[,pointer]...)
char *s, *format;

strtol Converts string to long integer.
long strtol(str,ptr,base)
char *str, **ptr;
int base;

toascii Returns argument with all but the 7 low order bits set to zero.
#include<ctype.h>
int toascii (c)
int c;

tolower
–tolower Converts characters to lowercase. _tolower has a smaller domain.
#include <ctype.h>
int tolower
int c;

toupper Converts characters to upper case.
#include <ctype.h>
int toupper (c)
int c;

ungetc Inserts character *c* into buffer.
#include <stdio.h>
int ungetc(c,stream)
char c;
FILE *stream;

unlink Removes a directory entry.
int unlink(path)
char *path;

write Writes nbyte bytes from buffer pointed to by buf to the file.
int write(fildes,buf,nbyte)
int fildes
char *buf;
unsigned int nbyte;

STRING OPERATIONS

#include <string.h>
char *s1, s2, *s, c;
int n;

char *strcat(s1,s2)	Appends s2 to end of s1.
char *strncat(s1,s2,n)	Appends max. of n chars. from s2 to s1.
int strcmp (s1,s2)	Compares s2 to s1 and returns an integer that is: 0 if s1 =s2 >0 if s1 >s2 <0 if s1 <s2
int strncmp(s1,s2,n)	Compares *n* chars of s2 to s1. Returns: 0 if s1 =s2 >0 if s1 >s2 <0 if s1 <s2
char *strcpy(s1,s2)	Copy s2 to s1.
char strncpy(s1,s2,n)	Copy n char from s2 to s1.
int strlen(s)	Return no. of chars. in s.
int index(s,c)	Move pointer to first *c* in *s*.
int rindex(s,c)	Move pointer to last *c* in *s*
char *xtrcat(s1,s2)	Append s2 to end of s1,
char *xtrcpy(s1,s2)	Copy s2 to s1. Return a pointer to end of s1.
char xtrncpy(s1,s2,n)	Copy n char from s2 to s1.

AUX. SERIAL PORT 2 FUNCTIONS

initporta Initializes Aux Serial Port 2.
#include "paval.h"
int initporta (stopbit, bitchar,bitrate, parity)
long stopbit, bitchar, bitrate, parity;

stopbit	ST1 = 1 stop bit
	ST15 = 1.5
	ST20 = 2

bitchar	DB5 = 5 data bits
	DB6 = 6
	DB7 = 7
	DB8 = 8

bitrate	F110 = 110 bps
	F300 = 300 bps
	F120 = 1200 bps
	F240 = 2400 bps
	F480 = 4800 bps
	F960 = 9600 bps
	F192 = 19200 bps

parity	PANO = No parity
	PAEV = Even parity
	PAOD = Odd parity

Returns:	0 = Successful –1 = Parameter error

sndpa Transmits data using Aux Serial Port 2.
#include "paval.h"
int sndpa (ptr, nb, timeout)
char *ptr;
long nb, timeout;
Returns: nb =No. of bytes transmitted
0 = Timeout
–1 Parameter error

recpa Receives data using Aux Serial Port 2.
#include "paval.h"
int recpa (ptr, timeout)
char *ptr;
long timeout;
Returns: nb =No. of bytes received
0 = Timeout
–1 Parameter error

rstdrv Flushes the driver reception buffer.
#include "paval.h"
int rstdrv()

C LIBRARY FUNCTIONS (CONTINUED)

WINDOW INTERFACE FUNCTIONS

assignleds Creates or changes LEDs.
assignLeds(vtnum, pleds, ledword)
long vtnum; Virtual terminal no.
long pleds; Points to 80 chars.
long ledword; Bits 0 – 9: If Bit i = 1, LED i+1 is on. If Bit i = 0 LED i+1 is off.
Bits 10–31 reserved.

closeform Changes from form to window mode.
closeform(vtnum)
long vtnum;

closevt Releases the virtual terminal.
closevt(***vtnum***)
long vtnum; Virtual terminal no.

disablecur Causes the cursor to be invisible
disablecur(***vtnum***)
long vtnum; Virtual terminal no.

enablecur Causes the cursor to be visible
enablecur(***vtnum***)
long vtnum; Virtual terminal no.

getch Gets a character from stdin.
char getch(vtnum)
long vtnum;

getcwt Waits for a character from stdin.
char getcwt(vtnum)
long vtnum;

openform Opens a form.
long openform()

openvt Assigns a virtual terminal.
long openvt(pname)
char pname; 23–character string

prndata Sends data to the printer.
prndata(data)
char *data;

putvt Displays a string on a virtual terminal.
*putvt(*vtnum,*"*string*")*
long vtnum; Virtual terminal no.
char *string; Maximum of 80 ASCII characters, VT100 format

selprn Selects the parameters for a printer.
selprn (*device, br, bits, sb, par*)
long device, br, bits, sb, par;
Default: Parallel, 9600, 8, 2, Even

device	0=Serial	*bits*	0=5 bits
	1=Parallel		2=6
br	3=300		1=7
	6=600		3=8
	12=1200	*sb*	1=1 stop bits
	24=2400		2=1.5
	48=4800		3=2
	96=9600	*par*	0=None
	192=19200		1=Odd
			3=Even

MATH LIBRARY: libm.a

```
#include <math.h>
double x,y;
int a, k;
```

log(x)	Base e logarithm. **double log(x)**
log10(x)	Base 10 logarithm. **double log10(x)**
log2(x);	Base 2 logarithm. **double log2(x)**
exp(x)	Base e exponential. **double exp(x)**
exp10(x)	Base 10 exponential. **double exp10(x)**
exp2(x)	Base 2 exponential. **double exp2(x)**
sin(x)	Transcendental functions. **double sin(x)**
cos(x)	**double cos(x)**
tan(x)	**double tan(x)**
asin(x)	Inverse transcendental functions. **double asin(x)**
acos(x)	**double acos(x)**
atan(x)	**double atan(x)**
sqr(x)	**double sqr(x)**
sqrt(x)	**double sqrt(x)**
powerd(x, y)	xy (equivalent to exp2(x*log2(y)) **double powerd(x, y)**
poweri(x,a)	xa (equivalent to exp2(x*log2(a)) **double poweri(x,a)**
dabs(x)	Absolute value (?x?). **double dabs(x)**
dint (x)	Integer part of the double that is the parameter. **int dint (x)**
mulpower2(x, k)	Fast floating point multiplication by 2k. **double mulpower2(x, k)**
lngamma(x)	Natural logarithym of the gamma function if 0<x<5.1X10305. **double lngamma(x)**
fac(k)	k!, where 0?k?170 **double fac(k)**

Matrix	**double matinv(a,c,n)**
	double *a;
	long *c;
	long n;

MS–DOS File Compatible Functions

Fmkdir Creates a directory.
#include <msfsuse.h>
Fmkdir(dirname)
char *dirname; (including path)

Frmdir Removes a directory.
#include <msfsuse.h>
Frmdir(dirname)
char *dirname; (including path)

Fsearch Searches for a file or directory.
#include <msfsuse.h>
Fsearch(name, option, rec)
char *name; (file/dir name and path)
int option; (0=first occ, 1 = next occ)
struct DREC *rec

vi COMMANDS

KEY

n	Number	*x*	Single upper or lower case character
opt	Option	CR	RETURN key (carriage return)
^	CTRL key	*pat*	Text and/or pattern matching characters

INPUT COMMANDS

a	Append after cursor.
A	Append at end of line.
i	Insert before cursor.
I	Insert before first non–blank.
o	Open and insert at line below.
O	Open and insert at line above.
^D	Backtab over autoindent.
0^D	Kill all autoindent.
^V	Insert a non–printing character.

DELETE COMMANDS

x	Delete character.
*n*x	Delete *n* characters.
X	Delete character before cursor.
dw	Delete word.
*n*dw	Delete *n* words.
dd	Delete line.
*n*dd	Delete *n* lines.
dt*x*	Delete to *x* in a line.
D	Delete rest of line.
d/*pat*CR	Delete up to *pat.*
d?*pat*CR	Delete back to *pat.*

INSERT–MODE COMMANDS

^W	Erase last word.
^H	Erase last character.
ERASE	Keyboard character (same as ^H)
KILL	Keyboard character. Kill input on current line.
ESC	Keyboard character. End insert mode..

MARKING COMMANDS

m*x*	Mark position with letter x.
'*x*	Return to mark x.
'*x*	Mark at first non–blank in line.

CHANGE COMMANDS

cw	Change word until ESC.
*n*cw	Change *n* words until ESC.
cc	Change line until ESC.
*n*cc	Change *n* lines until ESC.
ct*x*	Change to *x* until ESC.
C	Change rest of line until ESC.
c/*pat*CR	Change up to *pat.*
c?*pat*CR	Change back to *pat.*

GO TO (File) COMMANDS

G	Go to last line of file.
*n*G	Go to line *n.*
/pat	Go to next line matching *pat.*
?pat	Go to previous line matching *pat.*
n	Go to next / or ?.
N	Go to previous / or ?.
/pat/+n	Go to *n*th line after pat.
?pat?–n	Go to *n*th line before pat
''	Go to previous context.
''	Go to first non–blank in line.

INVOKE, EXIT, SAVE COMMANDS

vi *file*	Edit first line of *file.*
ZZ	Exit vi and save changes.
:w	Write (save) changes.
:w *file*	Write (copy) to *file.*
:q	Quit vi.
:wq	Write (save) changes and quit vi.
:q!	Quit vi without saving changes.

MODIFY COMMANDS

.	Repeat last operation.
~	Reverse case of letter.
J	Join lines.
<<	Shift line left 1 tab.
>>	Shift line right 1 tab.

vi COMMANDS (CONTINUED)

MOVE TO (Screen) COMMANDS

+	Move to 1st character, next line.
–	Move to 1st char. previous line.
↓ or j	Move to next line, same column.
↑ or k	Move to previous line.
→ or l	Move to the right.
← or h	Move to the left.
$	Move to end of line.

SCREEN ADJUSTMENT COMMANDS

^L	Clear and redraw screen.
^R	Retype, without deleted lines.
zCR	Redraw, current line at top.
z.	Redraw, current line at center.
z–	Redraw, current line at bottom.
/*pat*/z–	Redraw with *pat* line at bottom.
z*n*.	Redraw window with *n* lines.
^E	Scroll window down 1 line.
^Y	Scroll window up 1 line.
^F	Scroll forward 1 screen.
^B	Scroll backward 1 screen.
^D	Scroll down half a screen.
^U	Scroll up half a screen.

SEARCH PATTERNS COMMANDS

^	Beginning of line.
$	End of line.
.	Any character.
*	0 or more repetitions of character.
[*A–Z*]	Match any character from *A –Z*.
[*abc*]	Match *a, b* or *c*.
[^*abc*]	Match any char. except *a, b,c*.
\	Escape character for literal \ / $. ^ [' & * | ~
\<	Beginning of word.
\>	End of word.

SUBSTITUTION COMMANDS

s*text*	Substitute *text* for character.		
S	Substitute line.		
:s/*X*/*Y*/opt	Substitute first X with Y.		
	opt:	g	Change all occurrences
		c	Confirm each change.
		p	Print each change.
&	Repeat last :s request.		
:g/X/s//Y/opt	Globally substitute Y for X for first X on each line.		

UNDO COMMANDS

u	Undo last operation.
U	Restore current line.
"*n*p	Retrieve *n*th last delete.
"*n*1pu.u.u.	Scan previous *n* deletes.

WORD AND LINE COMMANDS

w	Move forward 1 word.
W	Move forward 1 word, including punctuation.
b	Move back 1 word.
B	Move back 1 word, including punctuation.
e	Move to end of word.
E	Move to end of word, including punctuation.
f*x*	Find *x* forward (to the right).
;	Repeat last f.
,	Reverse last f.

YANK AND PULL COMMANDS

yy or Y	Yank line to buffer.
*n*yy or *n*Y	Yank *n* lines to buffer.
p	Put lines back below cursor.
P	Put lines back above cursor.
"*x*y	Yank to buffer *x*.
"*x*p	Put from buffer *x*.

vi SOFTKEYS

FILE Softkeys

F1	Open	:e	F6	Read	:r
F2	Save	:w	F7	Set	:set
F3	Quit	:q	F8	Next	:n
F4	Sav/Qit	ZZ	F9	Rewind	:rew
F5	Revert	:e!	F10	EDIT Softkeys	

EDIT Softkeys

F1	Insert	i	F6	Paste	p
F2	Append	a	F7	Srch Fw	/
F3	Del chr	x	F8	Srch Bk	?
F4	Cut	Y	F9	Again	.
F5	Copy	Yp	F10	FILE Softkeys	

C PROTOCOL LIBRARY COMMON FUNCTIONS

These functions are available in all protocol libraries.

FLUSH Clears all outstanding frames in the reception buffer.
int flush()
Returns: 3 Receive buffer overflow

GETPHY Indicates physical interface setting.
int getphy()
Returns 2–byte integer as shown below.

Byte 0 Bit	7	6	5	4	3	2	1	0
Pin	105	108	140	141	104	103	114	115
Pin	4	20			3	2	15	17
Sig	RTS	DTR	SQ		RD	TD	SCT	SCR

Byte 1 Bit	7	6	5	4	3	2	1	0
Pin	106	107	109	122	125	142		
Pin	5	6	8	12	22			
Sig	CTS	DSR	CD	SDCD	RI			

GETPORT Identifies which port is communicating with the library.
int getport()
Returns: 0=Port A selected
1=Port B selected

GETIME Gets the number of milliseconds since the system was started.
#include <mtos –ux.h>
int getime(msbfr)
unsigned char *msbfr;

INITTIME Initializes the .01 and 1 second timers. Use initp1 to initialize the port before you use inittime.
int inittime()

P1RESET Restarts or resets the Front End Processor. Restart clears the reception buffer. Stop is similar to a hardware reset.
int p1reset(kind)
int kind; 0 Restart simulation
1 Stop simulation

SETLEDS Controls which port's LEDs are displayed on the front panel of a Dual Port machine.
int setleds(*port*)
int *port;* 0=Port A LEDS displayed
1=Port B LEDS displayed
Returns 0=Successful
1=Invalid parameter
2=Not a Dual Port machine

SETPHY Sets physical interface lines as below.
setphy()

DCE Bit	7	6	5	4	3	2	1	0
Pin	106	107	109	122	125	142		
Pin	5	6	8	12	22			
Sig	CTS	DSR	CD	SDCD	RI			

SETPHY (continued)

DTE Bit	7	6	5	4	3	2	1	0
Pin	105	108	140	141				
Pin	4	20						
Sig	RTS	DTR	SQ					

SETPORT Selects Port A or Port B for library.
int setport(port)
int *port;* 0=Port A
1=Port B
Returns 0=Successful
–1=Parameter out of range
–2=Port B not available

SETTIMER Sets the timer *value.*
settimer(*number,value*)
char *number;* 0=.01 timer (down)
1=.01 timer (up)
2=seconds (down)
3=seconds (up)
unsigned int *value;* timer setting
Returns: 0=Successful
1=Invalid *number*
2=inittime not performed

TIMER Returns the value of the timer.
int timer(*number*)
unsigned int number;
0=.01 (down)
1=.01 (up)
2=seconds (down)
3=seconds (up)

GLOBAL ERROR CODES

–1	Front End Processor (FEP) is busy
0	FEP is free
1	FEP is transmitting
–200	Port is busy
–201	FEP parameter
–201	FEP Parameter port
–202	Not valid on ISDN interface
–208	Code not found
–209	FEP cannot be started
–210	Application Invalid
–211	Invalid transmission mode
–212	Timeout
–213	No memory available
–214	FEP Code read
–215	FEP file not found
–216	FEP Code not loaded
–217	FEP halted
–218	No Port B
–219	Internal running
–220	FEP Load error
–222	Undefined status
–224	FEP Data not set

AUTO HDLC C LIBRARY FUNCTIONS

<u>AUTO HDLC LIBRARY FILENAME:</u> **libhdlc.a**

INITP1 Initializes P1 and loads software.
int initp1(*type1,type2,encode,bitrate*)
int *type1;* 0=DCE
1=DTE
2=ISDN
int *type2;* 0=Network
1=Subscriber
int *encode;* 0=NRZ
1=NRZI
unsigned long *bitrate;* 50 – 64000
Returns:
0=Successful
–1=Invalid parameter(s)
–2=P1 program not loaded
–3=Port is busy

RECEIVE Receives a frame from P1 and places it at address in *packet.*
char receive (*packet*)
char **packet;*
Returns:
0=Successful
1=Link not established
2=initp1 not performed

SET_N1 Sets N1 (maximum packet size).
int set_n1(*val*)
int *val;* Range: 1 – 512
Returns:
0=Successful
–1=Invalid value

SET_N2 Sets N2 (retransmissions).
int set_n2(*val*)
int *val;* Range: 1 – 255
Returns:
0=Successful
–1=Invalid value

SET_T1 Sets T1 timer.
int set_t1(val)
int *val;* Range: 1 – 255
Returns:
0=Successful
–1=Invalid value

SET_WINDOW
Sets window size.
int set_window(*val*)
int *val;* Range: 1 – 7
Returns:
0=Successful
–1=Invalid value

SLOF Disconnects link.
int slof()

SLON Establishes link by sending a SABM.
int slon()

STATUS Indicates status of frame level.
int status()
Returns:
0=Disconnected
1=Link connection requested
2=Frame reject
3=Link disconnection requested
4=Information transfer
5=Local station busy
6=Remote station busy
7=Local & remote stations busy

TRANSMIT Transmits I–frame.
*transmit(*packet, length*)*
char **packet;*
int length;
Returns:
0=Successful
1=P1 busy
2=initp1 not performed
3=Link not established

BOP C LIBRARY FUNCTIONS

BOP LIBRARY FILENAME: libbop.a

DISCARD Discards a frame prior to its entering a buffer.

init discard( )

Returns:

0 Frame discarded; no frame in buffer.

<0standard error codes

INITP1 Initializes P1. Loads simulation software. When initialized with this function, the maximum frame size handled by the simulator is 2 kbytes.

int initp1(*type, encode, bitrate, flag*)

int *type;* 0=DCE
1=DTE
2=ISDN

int *encode*; 0=NRZ
1=NRZI

unsigned long *bitrate;* 50 – 64000

int *flag;* 0=FF
1=7E

Returns:

0=Successful

–1=Invalid parameter(s)

–2=P1 program not loaded

GET_NXLEN Returns the length of next frame from FEP.

int get_nxlen ()

Returns:

0 = No new frame

>0 = length of next frame

<0 = Standard error codes.

GET_NXSTAT Gives status of next frame.

int get_nxstat ( )

Returns:

0 No new frame

1 Frame ok

2 Parity error in frame

3 Abort sequence in frame

<0Standard error codes.

INITP1_8K Initializes P1. Loads simulation software. When initialized with this function, the maximum frame size handled by the simulator is 8 kbytes. Monitoring applications cannot be run simultaneously when initialized with this function.

int initp_8k1(*type, encode,* bitrate, flag)

int *type;* 0=DCE
1=DTE
2=ISDN

int *encode*; 0=NRZ
1=NRZI

unsigned long *bitrate;* 50 – 64000

int *flag;*0=FF
1=7E

Returns:

0=Successful

–1=Invalid parameter(s)

–2=P1 program not loaded

RECEIVE Receives a frame from P1 and places it at address in *frame.*

char receive (*frame*)

char **frame;*

Returns:

0=Good CRC or no frame waiting

1=Bad CRC

2=initp1 not performed

3=Overflow

4=Abort frame received

SETFLG Changes the idle fill pattern.

int setflg(*flag*)

char *flag;* 0=FF
1=7E

TRANSMIT Transmits number of bytes in *length,* starting at address in *frame.*

int transmit(*mode, frame, length*)

int *mode;* 0=Good CRC
1=Bad CRC
2=Abort sequence

char **frame;*

int *length;*

Returns:

0=Successful

1=P1 busy

2=initp1 not performed

3=Parameter error

4=Buffer overflow

TREADY Returns status of P1 transmitter.

int tready()

Returns:

0=Transmitter ready

1=Transmitter not ready

2=initp1 not performed

3=Overflow

LAPD C LIBRARY FUNCTIONS

LAPD LIBRARY FILENAME: liblapd.a

GET_MOD Returns the current modulus.
int get_mod()
Returns:
0=Mod8
1=Mod128

GET_RNTEI Returns value of user–defined receive TEI.
int get_rntei(val)
int *val;* Range: 0 – 2
Returns:
0–127= TEI value
–1=*val* outside of range
–2= No extended memory

GET_RSAPI Returns value of user–defined SAPI.
int get_rsapi(val)
int *val;* Range: 0 – 2
Returns:
0 – 63= SAPI value
–1= *val* outside of range
–2= No extended memory

GET_SCONFIG Returns status configuration byte.
int get_sconfig
Returns status configuration byte (see manual for interpretation).

GET_SIM Returns the side being simulated.
int get_sim()
Returns:
0=Network
1=Subscriber

INITP1 Initializes P1 and loads software.
*int initp1(*interface,station,encode,bitrate*)*
int *interface;* 0=DCE
1=DTE
2=ISDN
int *station;* 0=Network
1=Subscriber
int *encode;* 0=NRZ
1=NRZI
long *bitrate;* 50 – 64000
Returns:
0=Successful
–1=Parameter error
–2=Code not found (see manual)
–3=Time out
–4=Can't set interface mode
–5=Can't set Vtype interface module
–6=Can't set bit rate
–7=Internal error
–8=Can't run FEP
–10=Can't restart simulator
–11=Can't initialize simulator
–13=Can't initialize timers

RECEIVE Receives an I–frame from P1 and places it at address in *rloc.*
extern int *rxlen*
int receive(*rloc*)
char *rloc;
Returns:
0=Successful or no frame waiting
2=*initp1* not performed
4=P1 busy

RESTARTSIM Restarts P1 simulation.
int restartsim()
Returns:
0=Successful
1=Time out

SETFLG Changes the idle fill pattern.
int setflg(flag)
int flag; 0 Fill with FF
1 Fill with 7E
Returns: 0 Successful
1 Timeout

SET_BIT_RATESets the bit rate.
int set_bit_rate(rate)
long *rate;* Range: 50–64000
Returns:
0=Successful
1=Error

SET_MOD Sets the modulus to mod8 or mod128
int set_mod(*val*)
int *val;* 0=Mod8
1=Mod128
Returns:
0=Successful
–1=*val* outside of range
1=Time out

S_N200 Sets N2 (retransmissions).
int s_n200(val)
int *val;*
Returns:
0=Successful
1=Time out

S_N201 Sets N1 (maximum packet size).
int s_n201(val)
int *val;* Range: 1 – 512
Returns:
0=Successful
–1=*val* outside of range
1=Time out

LAPD C LIBRARY FUNCTIONS – CONTINUED

SET_NET Sets simulation of network.
int set_net()
Returns:
0=Successful
1=Time out

SET_RNTEI Sets user–defined TEI value.
int set_rntei(*val*,tei)
int *val;* Range 0 – 2
int *tei;* Range 0 – 255
Returns:
0=Successful
1=Time out
–1=*val* or *tei* outside of range
–2=No extended memory

SET_RSAPI Sets value of user–defined SAPI.
int set_rsapi(*val, sapi*)
int *val;* Range 0 – 2
int *sapi;* Range 0 – 63
Returns:
0=Successful
1=Time Out
–1=*val* is outside range
–2=No extended memory

SET_SAPI Sets supported SAPI for transmission
int set_sapi(*val*)
int *val;* 0 – 63
Returns:
0=Successful
–1=*val* outside or range
1=Time out

SET_SCONFIG
Sets status configuration byte.
intset_sconfig(byte)
int *byte;*
Returns:
0=Successful
1=Time Out
–2=No extended memory

SET_SUB Sets simulation of subscriber.
int set_sub()
Returns:
0=Successful
1=Time out

S_T200 Sets value of the T200 timer.
int set_t1(*val*) *or* s_t200(val)
int *val;*
Returns:
0=Successful
1=Time out

S_T203 Sets value of T2 timer.
int set_t2(*val*) *or* s_t203(val)
int *val;*
Returns:
0=Successful
1=Time out

SET_TEI Sets the transmit TEI value.
int set_tei(*val*)
int *val;* Range: 0 – 127
Returns:
0=Successful
–1=*val* outside or range
1=Time out

SET_WINDOW
Sets window size.
int set_window(*val*)
int *val;* Range: 1 – 7
Returns:
0=Successful
–1=*val* outside of range
1=Time out

SLOF Disconnects link.
int slof()
Returns:
0=Successful
1=Time out

SLON Establishes link by sending a SABM or SABME.
int slon()
Returns:
0=Successful
1=Time out

LAPD C LIBRARY FUNCTIONS – CONTINUED

STATUS Indicates status of frame level.
int status()
Returns:
0=Disconnected
1=Link connection requested
2=Packet reject
3=Link disconnection requested
4=Information transfer
5=Local station busy
6=Remote station busy
7=Local and remote station busy
*8=Remote station not responding
other=LAPD not running

STOPSIM Stops P1 simulation.
int stopsim()
Returns:
0=Successful
1=Time out

TRANS Transmits a specified type of frame.
*int trans(*stat,frame,len)
int *stat;* 0x80=I–Frame
0x81=UI frame
0x82=XID Command frame
0x83=XID Response frame
char *frame;
int *len;* (0 – 511)
Returns:
0=Successful
1=P1 busy
2=initp1 not performed
3=Link not established
5=Time out
trxcni if an XID command with
len=0
trxrni if an XID response with
len=0

TRANSMIT Transmits number of bytes in *length,*
starting at address in *packet.*
int transmit(*packet, length*)
char *packet;
int *length;*
Calls and returns the value returned by:
trans(IFRAME,packet, length)

TRUI Transmits an unnumbered I–frame.
int trui(x*loc*,x*len*)
char *xloc; Location of data
int x*len;* Length of data field
Calls and returns the value returned
by: trans(UI,packet, length)

TRXCNI Transmits an XID command frame
without an I–field.
int trxcni()
Returns:
0=Successful
1=Time out

TRXIDC Transmits an XID command frame
with an I–field.
trxidc(x*loc*,x*len*)
char *xloc; Location in memory
int x*len;* Length
Calls and returns the value returned
by: trans(XIDC,xloc,xlen)

TRXIDR Transmits an XID response frame
with an I–field.
trxidr(x*loc*,x*len*)
char *xloc; Location in memory
int x*len;* Length
Calls and returns the value returned
by: trans(XIDR,xloc,xlen)

TRXRNI Transmits an XID response frame
without an I–field.
int trxrni()
Returns:
0=Successful
1=Time out

SDLC C LIBRARY FUNCTIONS

SDLC LIBRARY FILENAME: libsdlc.a

INITP1 Initializes P1 and loads software.
int initp1(*type1,type2,encode,bitrate*)
int *type1;* 0=DCE
1=DTE
2=ISDN
int *type2;* 0=Primary
1=Secondary
int *encode;* 0=NRZ
1=NRZI
unsigned long *bitrate;* 50 – 64000
Returns:
0=Successful
–1=Invalid parameter(s)
–2=P1 program not loaded

RECEIVE Receives a frame from P1 and places it at address in *packet.*
char receive (*packet*)
char **packet;*
Returns:
0=Successful
1=Link not established
2=initp1 not performed

SET_ADR Sets transmit and receive address
int set_adr(*val*)
char *val;* Range: 0 – 255
Returns:
0=Successful
–1=Parameter error

SET_N2 Sets N2 (number of retransmissions).
int set_n2(*val*)
int *val;* Range: 1 – 255
Returns:
0=Successful
–1=*val* outside of range
5=Not a primary station

SET_T1 Sets T1 timer.
int set_t1(val)
int *val;* Range: 1 – 255
Returns:
0=Successful
–1=*val* outside of range
5=Not a primary station

SET_T2 Sets T2 frame level timer.
int set_t2(val)
int *val;* Range: 0 – 255
Returns:
0=Successful
–1=*val* outside of range
5=Not a primary station

SLOF Disconnects link.
int slof()
Returns:
5=Not a primary station

SLON Establishes link by sending a SABM.
int slon()
Returns:
5=Not a primary station

STATUS Indicates status of frame level.
int status()
Chameleon 32 as Primary returns:
0=Normal disconnected mode
1=Link request state
2=Disconnect request state
3=Information Transfer state
4=Local station busy
5=Remote station busy
6=Local & remote stations busy

Chameleon 32 as Secondary returns:
0=Normal disconnected mode
1=Initialization mode
2=Frame reject
3=Information Transfer
4=Local station busy
5=Remote station busy
6=Local & remote stations busy

TRANSMIT Transmits I–frame with I–field of *length,* starting at address in *packet.*
int transmit(*packet, length*)
char **packet;*
int length;
Returns:
0=Successful
1=P1 busy
2=initp1 not performed
3=Link not established
4=Length error (if length > 510)

TRNSI Transmits a non–sequenced I–frame with I–field of *length,* starting at address in *packet.*
int trns*i (packet, length)*
char **packet;*
int *length;*
Returns:
0=Successful
1=P1 busy
2=initp1 not performed
3=Link not established
4=Length error (if length > 510)

SDLC C LIBRARY FUNCTIONS – CONTINUED

TRSIFR Transmits a sequenced I–frame with I–field of *length*, starting at address in *packet*.
int trsifr(*packet, length*)
char **packet;*
int *length;*
Returns:
0=Successful
1=P1 busy
2=initp1 not performed
3=Link not established
4=Length error (if length > 510)

TRTST Transmits a test frame with I–field of *length*, starting at address in *packet*.
int trtst (*packet, length*)
char **packet;*
int *length;*
Returns:
0=Successful
1=P1 busy
2=initp1 not performed
3=Link not established
4=Illegal frame (if secondary)
5=Not a primary station

TRUI Transmits an unnumbered I–frame with I–field of *length*, starting at address in *packet*.
int trui (*packet, length*)
char **packet;*
int *length;*
Returns:
0=Successful
1=P1 busy
2=initp1 not performed
3=Link not established
4=Length error (if length > 510)

XID Transmits an XID frame containing the data in the externally available character array *ident[ ]*.
extern char ident[ ]; /* 6 bytes */
char xid();
Returns:
0=Successful
1=P1 not initialized
2=P1 fails to respond
3=Not in normal response mode
4=Illegal frame (if secondary)

BASIC RATE INTERFACE LIBRARY FUNCTIONS

FILENAME: libbri.a

SetBasic int SetBasic(cmdblock, resblock);
int cmdblock [5];
int resblock [5];

The error codes for resblock[0] for all Basic Rate Library commands and are listed below.

resblock [0]	Meaning
00	Successful
01	Hardware has already been set up
02	Requested function is not available for this configuration
03	Requested channel is invalid (for B1, B2 and D)
04	Requested function is not available for this channel
05	Invalid command or request
10	Basic Rate Interface board is not installed

Setup	cmdblock[0] = 1 (Board 0) or cmdblock[0] = 101 (Board 1)			
	cmdblock[1]	mode	*1*	*Monitor*
			2	Simulate NT
			3	Simulate TE
	resblock[1]	Returns current mode, if unsuccessful		
Reactivate	cmdblock[0] = 2 (Board 0) or cmdblock[0] = 102 (Board 1)			
	Argument	None		
Reset	cmdblock[0] = 3 (Board 0) or cmdblock[0] = 103 (Board 1)			
	Argument	None		
Channel Functions	cmdblock[0] = 4 (Board 0) or cmdblock[0] = 104 (Board 1)			
	cmdblock[1]	*mode*	0	If request conflicts with current setup, do not override.
			1	If request conflicts with current setup, override.
	cmdblock[2]	*channel*	1	B1 channel
			2	B2 channel
			3	D channel
	cmdblock[3]	*selection*	1	System
			2	Milliwatt
			3	Codec
			4	External interface
			5	Idle
	resblock[1]	*channel* as defined above (If resblock[0] 0)		
	resblock[2]	*selection* as defined above (If resblock[0] 0)		
Signal Functions	cmdblock[0] = 5 (Board 0) or cmdblock[0] = 105 (Board 1)			
	cmdblock[1]	For NT	1	Deactivate request
			2	Send info–2
			3	Send info–4
			4	Activate NT
			5	Reserved
			6	Send single pulses
			7	Send continuous pulses
			8	Send info–2, test loop 2
			9	Send info–4, test loop 2
		For TE	1	Deactivate
			2	Activate at priority 8
			3	Activate at priority 10
			4	Activate TE
			5	Reserved
			6	Reserved
			7	Reset PEB 2080
			8	Send single pulses
			9	Send continuous pulses
			10	Activate test loop 3

BASIC RATE INTERFACE LIBRARY FUNCTIONS – CONTINUED

Get Status	cmdblock[0] = 6 (Board 0) or cmdblock[0] = 106 (Board 1)		
	Argument	None	
	resblock[1]	Control byte received from PEB 2080.	
	resblock[2] (If Simulating an NT):		
		1	No clock signal
		2	Lost signal level
		3	Receiver not synchronous
		4	Error
		5	Info–1 received
		6	Receiver synchronized
		7	Deacitvation complete
		8	Undefined
	resblock[2] (If Simulating a TE):		
		1	Power up
		2	Deactivate request
		3	Slip detected
		4	Disconnected
		5	Error
		6	Resynchronizing
		7	Info–2 received
		8	Test mode
		9	Level received during test loop
		10	Info–4 received, D channel priority 8 or 9
		11	Info–4 received, D channel priority 10 or 11
		12	Quiescent state
		13	Undefined
	If in Monitor mode:		
	resblock[1]	Control byte received from PEB 2080.	
	resblock[2]	Same as resblock[2] from NT	
	resblock[3]	Same as resblock[2] from TE	
Select Trace Option	cmdblock[0] = 9 (Board 0 only)		
	cmdblock[1]	0	Turns off the trace
		1	Command/result display
		2	Detailed trace
NT Power	cmdblock[0] = 10		
	cmdblock[1] mode	1	Power source 1 (normal conditions)
		2	Power source 1 (emergency conditions)
		3	Power source 2 (normal conditions)
		4	Power source 2 (emergency conditions)
		5	Off
Bas_version	This function returns the current version of the BRI library. **char *Bas_version()**		

2B1Q U–INTERFACE C LIBRARY FUNCTIONS

FILENAME: libu.a

SetU

```
int SetU(cmdblock, resblock);
char cmdblock [ ];
char resblock [ ];
```

The error codes for resblock [0] for all U–Interface Library commands are listed below.

resblock [0]	Meaning
00	Successful
01	Invalid command
02	Invalid command parameters
03	Requested board is not responding
04	U–board physical error
05	U–board interface is not initialized
06	Requested board is not installed

```
Initialize     cmdblock[1]  =  0 (Board 0)   or cmdblock [1]  =  100 (Board 1)
Configure      cmdblock[0]  =  1 (Board 0)   or cmdblock [0]  =  101 (Board 1)
               cmdblock[1]  mode 1      Monitor
                                 2      Simulate NT
                                 3      Simulate TE
               resblock[1]  Returns current mode, if unsuccessful
Set
Transceiver
State          cmdblock[0]  =  2  (Board 0)  or cmdblock[0]  =  102  (Board 1)
               cmdblock[1]  =  Xcvr specifier  0  =  NT Xcvr
                                               1  =  LT Xcvr
               cmdblock[2]  =  Xcvr State      1  =  Reset
                                               2  =  Power down
                                               3  =  Absolute
                                               4  =  Normal
Get
Transceiver
State          cmdblock[0]  =  3  (Board 0)  or cmdblock[0]  =  103  (Board 1)
               cmdblock[1]  =  Xcvr specifier  0  =  NT Xcvr
                                               1  =  LT Xcvr
               cmdblock[2]  =  Xcvr State      1  =  Reset
                                               2  =  Power down
                                               3  =  Absolute
                                               4  =  NormalSet
Transceiver
Activation     cmdblock[0]  =  4  (Board 0)  or cmdblock[0]  =  104  (Board 1)
               cmdblock[1]  =  Xcvr specifier  0  =  NT Xcvr
                                               1  =  LT Xcvr
               cmdblock[2]  =  Xcvr State      1  =  Start activation
                                               2  =  Start deactivation
Get
Transceiver
Connection     cmdblock[0]  =  5  (Board 0)  or cmdblock[0]  =  105  (Board 1)
               cmdblock[1]  =  Xcvr specifier    0  =  NT Xcvr
                                                 1  =  LT Xcvr
               resblock[0]  =  See Error Code
               resblock[1]  =  Xcvr Connection   0  =  None
                                                 1  =  Port A
                                                 2  =  Port B
                                                 3  =  Ports A and B
```

2B1Q U–INTERFACE C LIBRARY FUNCTIONS (continued)

Set Transceiver Errors	cmdblock[0]	= 6 (Board 0) or cmdblock[0] = 106 (Board 1)
	cmdblock[1]	= Xcvr specifier 0 = NT Xcvr
		1 = LT Xcvr
Get Transceiver Errors	cmdblock[0]	= 7 (Board 0) or cmdblock[0] = 107 (Board 1)
	cmdblock[1]	= Xcvr specifier 0 = NT Xcvr
		1 = LT Xcvr
	resblock[0]	= See Error Codes
	resblock[1–4]	= 32–bit FEBE count. MSBs followed by LSBs.
	resblock[5–8]	= 32–bit NEBE count. MSBs followed by LSBs.
	resblock[9–12]	= 32–bit NoSyn count. MSBs followed by LSBs.
	resblock[13–16]	= 32–bit NoAct count. MSBs followed by LSBs.
Get HW Version	cmdblock[0]	= 8 (Board 0) or cmdblock[0] = 108 (Board 1)
	resblock[0]	= See Error Codes
	resblock[1]	= NT xcvr version number.
	resblock[2]	= LT xcvr version number.
Get Link Status	cmdblock[0]	= 9 (Board 0) or cmdblock[0] = 109 (Board 1)
	resblock[0]	= See Error Codes
	resblock[1]	= NT link status
		bit 0 = link up
		bit 1 = superframe sync recognized
		bit 2 = Xcvr activation in progress
		bit 3 = error indicator
	resblock[2]	= LT link status
		bit 0 = link up
		bit 1 = superframe sync recognized
		bit 2 = Xcvr activation in progress
		bit 3 = error indicator
Transceiver Transmit	cmdblock[0]	= 11 (Board 0) or cmdblock[0] = 111 (Board 1)
	cmdblock[1]	= Xcvr specifier 0 = NT Xcvr
		1 = LT Xcvr
	cmdblock[2]	= Channel specifier
		1 = EOC
		2 = M4
		3 = M5/M6
	cmdblock[3]	= EOC address, EOC DM bit
	cmdblock[4]	= EOC information
	cmdblock[5]	= M4 information
	cmdblock[6]	= M5/M6 information
Transceiver Receive	cmdblock[0]	= 12 (Board 0) or cmdblock[0] = 112 (Board 1)
	cmdblock[1]	= Xcvr specifier 0 = NT Xcvr
		1 = LT Xcvr
	resblock[0]	= See Error Codes
	resblock[1]	= Message Length 0 = No data available
		1 = 6 data bytes follow
		2 = M4
		3 = M5/M6

```
                resblock[2]  =  EOC address, EOC DM bit
                resblock[3]  =  EOC information
                resblock[4]  =  EOC address, EOC DM bit
                resblock[3]  =  EOC information
                resblock[6]  =  M4 information
                resblock[7]  =  M5/M6 information


EOC
Processing      cmdblock[0]  =  13  (Board 0)  or cmdblock[0]  =  113  (Board 1)
                cmdblock[1]  =  Xcvr specifier  0  =  NT Xcvr
                                                1  =  LT Xcvr
                cmdblock[2]  =  Automatic processing mode
                                                1  =  No action
                                                2  =  Operate 2B + D Loopback
                                                3  =  Operate B1 Loopback
                                                4  =  Operate B2 Loopback
                                                5  =  Send corrupted CRC
                                                6  =  Return to Normal
EOC Mode
Control         cmdblock[0]  =  14  (Board 0)  or cmdblock[0]  =  114  (Board 1)
                cmdblock[1]  =  Xcvr specifier  0  =  NT Xcvr
                                                1  =  LT Xcvr
                cmdblock[2]  =  EOC Reception mode
                                                1  =  No action
                                                2  =  Handle every EOC
                                                3  =  Handle EOC passing trinal checks
                                                4  =  Handle EOC passing trinal checks
                                                      with automatic EOC processing
M4 Mode
Control         cmdblock[0]  =  15  (Board 0)  or cmdblock[0]  =  115  (Board 1)
                cmdblock[1]  =  Xcvr specifier  0  =  NT Xcvr
                                                1  =  LT Xcvr
                cmdblock[2]  =  M4 Reception mode
                                                0  =  No action
                                                1  =  Handle dual–consecutive M4 with
                                                      verified act/dea
                                                2  =  Handle dual–consecutive M4
                                                3  =  Handle Delta M4
                                                4  =  Handle every M4
M5/6 Mode
Control         cmdblock[0]  =  16  (Board 0)  or cmdblock[0]  =  116  (Board 1)
                cmdblock[1]  =  Xcvr specifier  0  =  NT Xcvr
                                                1  =  LT Xcvr
                cmdblock[2]  =  M4 Reception mode
                                                0  =  No action
                                                1  =  Handle dual–consecutive M5/6
                                                3  =  Handle Delta M5/6
                                                4  =  Handle every M5/6
Shutdown        cmdblock[0]  =  30  (Board 0)  or cmdblock[0]  =  130  (Board 1)
```

BSC C LIBRARY FUNCTIONS

<u>BSC LIBRARY FILENAME:</u> libbsc.a

IDLE_MODE Specifies the character to be transmitted while the line is idle.

```
#include <chamh>
int idle_mode(mode)
int mode;
        IDLE or 0     FF is transmitted
        SYNC or 1     SYN is transmitted
```

INITP1 Initializes P1. Loads simulation software.

```
int initp1(type,encode, bitrate,crc, data)
int type;        0=DCE
                 1=DTE
int data;   0x10    EBCDIC data
            0x04    ASCII (no parity)
            0x01    ASCII (even parity)
            0x00    ASCII (odd parity)
struct control encode
```

Defines the control characters for BSC, as follows:

```
     struct control
        {
        unsigned char eot;
        unsigned char syn;
        unsigned char dle;
        unsigned char stx;
        unsigned char etx;
        unsigned char soh;
        unsigned char etb;
        unsigned char itb;
        unsigned char enq;
        };
     unsigned long bitrate; 50 – 64000
     char crc;     0=CRC16
                   1=CCITT–CRC
     Returns:
       0=Successful
      –1=Invalid parameter(s)
      –2=P1 program not loaded
```

RECEIVE Receives a frame from P1 and places it at address in *frame*.

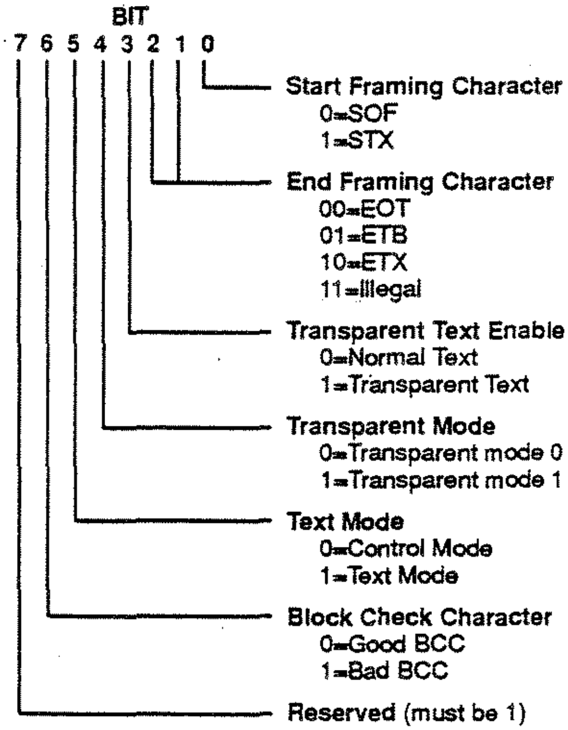
```
char receive (frame)
  char *frame;
Returns:
 0=Good BCC or no frame waiting
 1=Bad BCC
 2=initp1 not performed
 3=Overflow
```

TREADY Returns status of P1 transmitter.

```
int tready()
Returns:
  0=Transmitter ready
  1=Transmitter not ready
  2=initp1 not performed
  3=Overflow
```

TRANSMIT Transmits number of bytes in *length*, starting at address pointed to by **frame*, with the control characters and BCC as specified by *mode*. *Mode* is bit encoded as shown in the figure below.

```
int transmit(mode, frame, length)
char *frame;
int length;
char mode;
Returns:
   0=Successful
   1=P1 busy
   2=initp1 not performed
   3=Parameter error
   4=Buffer overflow
```



C PRIMARY RATE INTERFACE LIBRARY FUNCTIONS

FILENAME: libpri.a

SetPrimary

```
int SetPrimary(cmdblock, resblock);
int cmdblock [14];
int resblock [14];
```

The error codes for resblock[0] for all Primary Rate Interface Library commands are:

resblock[0]	Meaning
0	Successful
1	Primary Rate Interface board is not installed
2	Setup already done
3	Invalid channel number/time slot
4	Selection already in use
5	Channel already assigned
10	Command not implemented

Setup cmdblock[0] = 1 (Board A) or cmdblock[0] = 101 (Board B)

cmdblock[1]	*mode*	1	Monitor
		2	Simulate
cmdblock[2]	*framing*	1	D4
		2	ESF
		3	SL96
		4	CEPT
cmdblock[3]	*idle data*		8 bit value
cmdblock[4]	*idle signal*		2 or 4 bit value
*cmdblock[5]	*DS0x receive*		Channel/time slot
*cmdblock[6]	*Codec receive*		Channel/time slot
*cmdblock[7]	*DS0y receiver/transmitter*		Channel/time slot
*cmdblock[8]	*Codec transmitter*		Channel/time slot (Ignored in Monitor mode)
*cmdblock[9]	*Milliwatt transmitter*		Channel/time slot (Ignored in Monitor mode)
*cmdblock[10]	*Status line 1*		one byte (See Setup byte interpretation on next page.)
cmdblock[11]	*Status line 2*		one byte (See Setup byte interpretation on next page.)

* Available for line 1 only.

Resynchronize cmdblock[0] = 2 (Board A) or cmdblock[0] = 102 (Board B)
Argument None

Reset cmdblock[0] = 3 (Board A) or cmdblock[0] = 103 (Board B)
Argument None

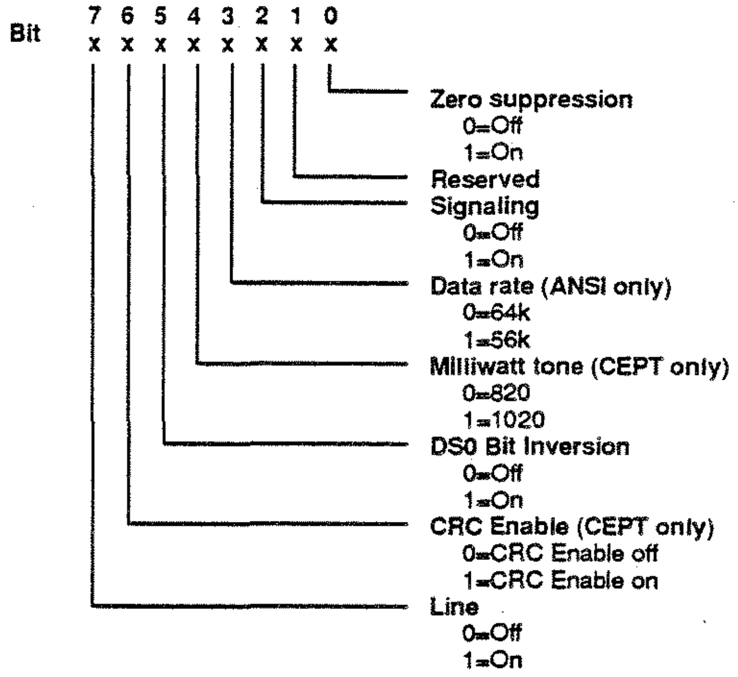
Channel Functions cmdblock[0] = 4 (Board A) or cmdblock[0] = 104 (Board B)

cmdblock[1]	*mode*	0	If request conflicts with current setup, retain current setup.
		1	If request conflicts with current setup, override current setup.
cmdblock[2]	*selection*	1	DS0x receive
		2	Codec receive
		3	DS0y transmit
		4	DS0y receive
		5	Codec transmit
		6	Milliwatt transmit
		7	Reset transmit channel
		8	Reset receive channel
		9	Idle data
		10	Idle signal
cmdblock[3]	*channel (if cmdblock[2]=1 – 8)*		1 – 24 D4/ESF line 1 1 – 31 CEPT line 1
cmdblock[3]	*idle bits (if cmdblock[2]=9 – 10)*		8, 4, 2, bits

Reserved cmdblock[0] = 5 (Board A) or cmdblock[0] = 105 (Board B)
Reserved for future use.

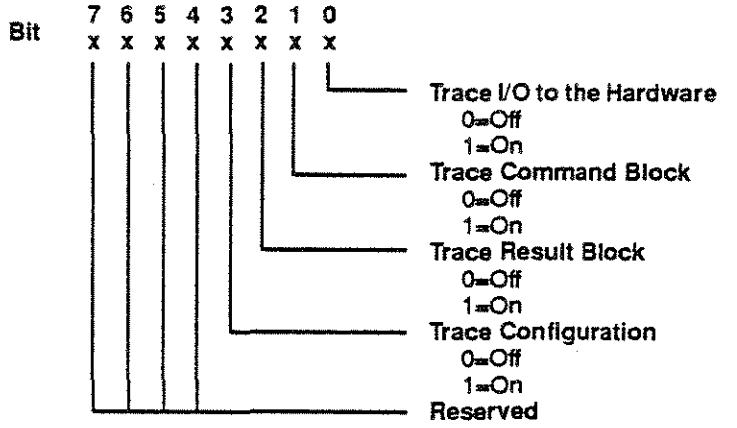
Get Status cmdblock[0] = 6 (Board A) or cmdblock[0] = 106 (Board B)
Argument None
Returns the current configuration in the Setup command configuration format.

Change Status cmdblock[0] = 7 (Board A) or cmdblock[0] = 107 (Board B)
cmdblock[1] *line 1 or 2*
cmdblock[2] *status 8 bits*, interpreted as follows:



Zero suppression
0=Off
1=On
Reserved
Signaling
0=Off
1=On
Data rate (ANSI only)
0=64k
1=56k
Milliwatt tone (CEPT only)
0=820
1=1020
DS0 Bit Inversion
0=Off
1=On
CRC Enable (CEPT only)
0=CRC Enable off
1=CRC Enable on
Line
0=Off
1=On

Enable Trace cmdblock[0] = 9 (Board A only)
cmdblock[1]8 **trace bits**, interpreted as follows:



Trace I/O to the Hardware
0=Off
1=On
Trace Command Block
0=Off
1=On
Trace Result Block
0=Off
1=On
Trace Configuration
0=Off
1=On
Reserved

Pri_version Returns the current verion of the PRI library.
char *Pri_version();

ASYNC LIBRARY QUICK REFERENCE

ASYNC LIBRARY FILENAME: libasc.a

INITP1 Initializes P1 and loads simulation software.

int initp1(*type, encode*)

int *type;* 0=DCE
1=DTE

struct ASC_CTRL **encode*

```
struct ASC_CTRL
    {
    int bitrate;
    int parity;
    int stop;
    int data;
    int duplex;
    int block;
    int eob;
    };
```

bitrate	1	50	7	1200
	2	75	8	2400
	3	110	9	4800
	4	150	10	9600
	5	300	11	19200
	6	600		

parity	0	None
	1	Odd
	2	Even
stop	0	1 Stop bit
	1	1.5 Stop bits
	2	2 Stop bits
data	5	5 Data bits
	6	6 Data bits
	7	7 Data bits
	8	8 Data bits
duplex	0	Full duplex
	1	Half duplex
block	0	Block mode
	1	Character mode

eob (End of block character): 0–0xFF

Returns:
0=Successful
–1=Invalid parameter(s)
–2=P1 program not loaded
–3=Port is busy

RECEIVE Receives a block or character from P1 and places it at address pointed to by *frame.*

char receive (*frame*)

char **frame;*

Returns:
0=Good BCC or no frame waiting
1=Bad BCC
2=initp1 not performed
3=Overflow

TBREAK Transmits a break sequence.

int tbreak()

TRANSMIT Transmits number of bytes in *length,* starting at address pointed to by **frame,* with the control characters and BCC as specified by *mode.*

int transmit(*frame, length*)

char **frame;*

int *length;*

Returns:
0=Successful
1=P1 busy
2=initp1 not performed
3=Parameter error
4=Buffer overflow

TREADY Returns status of P1 transmitter.

int tready()

Returns:
0=Transmitter ready
1=Transmitter not ready
2=initp1 not performed
3=Overflow

C ANALYSIS LIBRARY FUNCTIONS

FILENAME: libanal.a

init_anal This function initializes the hardware and loads the analysis software.

```
int init_anal(port, protocol, par)
int port, protocol;
union PARBLOCK *par;
```

Port	0	Port A
	1	Port B
	2	Port A and B
Protocol	1	BOP
	2	ISDN
	7	Async
	8	BSC

Par:

```
union PARBLOCK  {          /* BOP parameter block */
          struct {
              unsigned short nrz;
          } pbop;

          struct   {               /* Bisync parameter block */
              unsigned short  table;
              unsigned short  bsc;
              charsync1;
              charsync2;
              unsigned short  parity;
          }pbisync;

          struct {               /* Async parameter block */
              unsigned short  baud;
              unsigned short  parity;
              unsigned short  databit;
          }pasync;
};
```

BOP and ISDN: If Protocol = 1 (BOP) or 2 (ISDN), the following parameter must be initialized:

par—>pbop.encode	0	NRZ
	1	NRZI

ASYNC: If Protocol = 7 (Async), the following three parameters must be initialized:

par—>pasync.baud2	75	baud rate
	3	110
	5	300
	6	600
	7	1200
	8	2400
	9	4800
	10	9600
	11	19200
par—>pasync.parity	0	None
	1	Odd
	2	Even
par—>pasync.databit	5	5 data bit
	6	6 data bits
	7	7 data bits
	8	8 data bits

BSC If Protocol=8 (BSC), the following parameters must be initialized:

par–>pbsync.table	0	ASCII
	1	EBCDIC
par–>pbsync.bcc	0	CRC16
	1	LRC
	2	CCITT
par–>pbsync.sync1	Range:	0 – 0xff
par–>pbsync.sync2	Range:	0 – 0xff

AND if par–>pbsync.table is initialized to ASCII the following parameter must also be initialized:

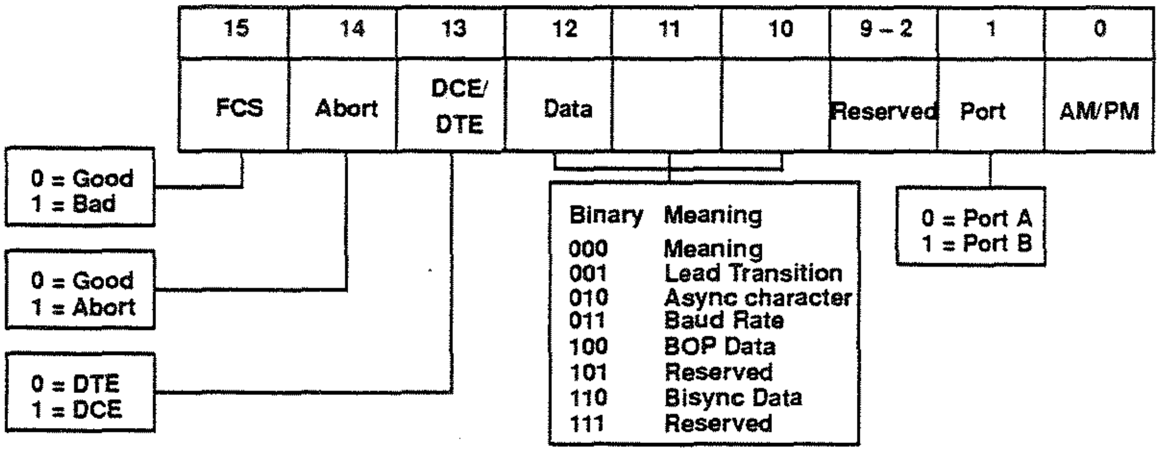
par–>pbsync.parity	0	None
	1	Odd
	2	Even

Returns

0	Successful
–1	Parameter error
–2	Dual ports not available
–3	Cannot load analysis files.
–4	Simulation is running
–5	Port is busy

getevent This function gets an event from the line, if available. Event is a special data type definition which is defined in a:\include\cham.h. It is defined as follows:

```
typedef struct {
  unsigned short type;      /*event.type     Bit–mapped information element (see figure
                                             below)*/
  unsigned short length;    /*event.length   The length of the data */
  unsigned short buflen;    /*event.buflen   Data buffer length*/
  unsigned char  *pdata;    /*event.pdata    Data buffer address that points to the frame*/
  long seconds;             /*event.secondsNumber of seconds since midnight or noon*/
  long ms20;                /*event.ms20     Number of 20 microseconds since the second*/
  unsigned short special;   /*event.special  If a baud rate event, the baud rate change event
                                             contains the new baud rate value. If a lead
                                             transition event, the bits indicate the lead states.
                                             */
  unsigned short crc;       /*event.crc      The crc of the frame*/
  unsigned short flags;     /*event.flags    For BOP only, contains the number of flags*/
} event;
#include <cham.h>
int getevent(pevent)
event *pevent;
Returns 0   Successful
       –1   No new events
       –2   Data overwitten (buffer wrapped)
```



15	14	13	12	11	10	9 – 2	1	0
FCS	Abort	DCE/ DTE	Data			Reserved	Port	AM/PM

Binary	Meaning
000	Meaning
001	Lead Transition
010	Async character
011	Baud Rate
100	BOP Data
101	Reserved
110	Bisync Data
111	Reserved

reset_anal Resets the acquisition processor.
int reset_anal()

MULTI–LINK LAPD LIBRARY QUICK REFERENCE

MULTI–LINK LAPD LIBRARY Filename: libmlapd.a

find_link
Returns the number of the lowest link matching the SAPI/TEI/TGE value specified.
int find_link(sapi,tei,tgi)
int sapi, tei, tgi;
Returns:
0 – 63 Matching link number
–1 No match found

get_freelink
Gets the number of first disabled link.
int get_freelink()
Returns:
0 – 63 Disabled link number
–1 No free links
–2 initp1 not performed

get_fwaiting
Gets the number of I–frames waiting to be transmitted on the link.
int get_fwaiting (lnkn)
char lnkn; 0 – 63
Returns: 0 – 7 No. of I–frames

get_link
Gets the number of the link currently under user control.
int get_link()
Returns:
0 – 63 Current link no.
–1 initp1 not performed

get_lnksapi
Gets the SAPI value for linkn.
int get_lnksapi (lnkn)
char lnkn; 0 – 63
Returns:
0 – 63 SAPI value
> 63 Disabling SAPI value

get_lnktei
Gets the TEI value for link lnkn.
int get_lnktei (lnkn)
char lnkn; 0 – 63
Returns:
0 – 127 TEI value
> 127 DisablingTEI value

get_lnktgi
Gets the TGI value for link lnkn.
int get_lnktgi (lnkn)
char lnkn; 0 – 63
Returns:
0 – 14 TGI value
15–255 Disabling TGI value

get_meswaiting
Gets no. of messages waiting to be received from the FEP.
int get_meswaiting ()
Returns: 0 – 32 No. of msgs.

get_rlink
Gets the number of the link which sent the last received message.
int get_rlink()
Returns:
0 – 63 Current link no.
–1 No messages rec'd
–2 initp1 not performed

get_rntei
Dummy function to maintain compatibility with single link LAPD programs that are being upgraded to Multi–Link LAPD.
int get_rntei (val)
int val;

get_rsapi
Dummy function to maintain compatibility with the existing single link LAPD programs that are being upgraded to Multi–Link LAPD.
int get_rsapi (val)
int val;

get_rxstat
Gets the low order byte of the frame status byte frstat for the last received message.
char get_rxstat()
Returns:
0–0xC3 frstat value
0xFF No messages recd
0xFE initp1 not performed

get_sapi
Gets the SAPI value of the link currently under user control.
int get_sapi()
Returns: 0 – 255 SAPI value

get_sconfig
Returns a copy of the current control configuration byte.
int get_sconfig ()

get_sim
Returns a copy of of the network/ subscriber selection.
int get_sim ()
Returns: 0 Network
1 Subscriber

get_tei
Gets the TEI of the link currently under user control.
int get_tei()
Returns: 0 – 255 TEI value

get_tgi
Gets the TGI of the link currently under user control.
int get_tgi()
Returns: 0 – 14 TGI value
15–255 Disabling TGI value

get_window
Gets the number of outstanding I–frames on link number lnkn.
int get_window (lnkn)
char lnkn; 0 – 63
phoneReturns: 0 – 7 No. of I–frames

MULTI–LINK LAPD LIBRARY (CONTINUED)

initp1 initp1 loads the Front End Process (FEP) code for the selected library and starts simulation. This is the same as the start_sim function, but is included for downward compatibility with the single link LAPD library.
int initp1 (interface, sta, encode, bitrt)
int interface, sta, encode;
long bitrt;

interface	0	V–type interface (DCE)
	1	V–type interface (DTE)
	2	ISDN interface
sta	0	Network
	1	Subscriber
encode	0	NRZ
	1	NRZI
bitrt	50 – 64000	

link_stat Gets the current state of link n.
int link_stat(n)
char n; 0 – 63
Returns: 0 – 9 Current state

0	Link Disconnected
1	Link Connection Requested
2	Frame Rejected
3	Disconnect Requested
4	Information Transfer
5	Local Station Busy
6	Remote Station Busy
7	Local & Remote Station Busy
8	Remote Stn not Responding
9	Link Disabled

receive Receives a message from the FEP
int receive(dest_addr)
char *dest_addr;

s_n200 Sets maximum number of retries (N200).
int s_n200 (val)
int val; 1 – 255
Returns: 0 Successful

s_n201 Sets maximum length of an I–frame (N201).
int s_n201 (val)
int val; 1 – 512
Returns: 0 Successful

s_t200 Sets the time allowed for the remote station to respond (T200). Setting this value to 0 disables the T200 timer.
int s_t200 (val)
int val; 0 – 255
Returns: 0 Successful

s_t203 Sets the maximum time between frames (T203). Setting this value to 0 disables the T203 timer.
int s_t203 (val)
int val; 0 – 255
Returns: 0 Successful

set_link Puts link n under user control. Only one link at a time can be under user control.
int set_link(n)
char n; 0 – 63

Returns:	0	Successful
	–1	Parameter error
	–2	initp1 not performed
	–3	Timeout

set_net Sets simulation side to network.
int set_net ( )

set_rntei This is a dummy function to maintain compatibility with existing LAPD link programs that are being upgraded to Multi–Link LAPD.
int set_rntei (val, tei)
int val, tei;

set_rsapi This is a dummy function to maintain compatibility with existing LAPD programs that are being upgraded to Multi–Link LAPD.
int set_rsapi (val, sapi)
int val, sapi;
This function always returns zero.

set_sapi Sets the SAPI value for the link under user control.
int set_sapi(v)
char v;
Accepted range of *v* is 0 – 255. A value over 63 will disable the selected link.

Returns:	0	Successful
	–1	Parameter out of range
	–2	initp1 not performed
	–3	Timeout

set_sconfig Sets the value of the control configuration byte
int set_sconfig (byte)
int byte;
Returns: 0 Successful

set_sub Set the simulation side to Subscriber.
int set_sub ( )

set_tei Sets the TEI value for the link under user control.
int set_tei(v)
char v; 0 to 255
> 127 diables link

Returns:	0	Successful
	–1	Parameter error
	–2	initp1 not performed
	–3	Timeout

MULTI–LINK LAPD LIBRARY QUICK REFERENCE (CONTINUED)

set_tgi Sets the TGI value for the link under user control.
int set_tgi(v)
char v; 0 to 14 TGI value
15 – 255 Diables use of TGI
Returns: 0 Successful
–1 Parameter error
–2 initp1 not performed
–3 Timeout

set_window Sets the maximum number of outstanding frames on each link.
int set_window (val)
int val; 1 – 7
Returns: 0 Successful

setflg Selects an interframe fill pattern.
int setflg (flag)
int flag; 0 0x7E fill
1 0xFF fill
Returns: 0 Successful

slof Disconnects the link.
int slof ()
Returns: 0 Successful

slon Attempts to establish a link.
int slon ()
Returns: 0 Successful

srch_lnk Returns the number of lowest link matching the specified SAPI/TEI.
int srch_lnk(sapi,tei)
Returns: 0 – 63 No. of lowest link
–1 No match found

start_sim start_sim loads the Front End Process (FEP) code for the selected library and starts simulation. (Identical to initp1 function.)
int start_sim (interface, sta, encode, bitrt)
int interface, sta, encode;
long bitrt;
interface 0 V–type interface (DCE)
1 V–type interface (DTE)
2 ISDN interface
sta 0 NETWORK
1 SUBSCRIBER
encode 0 NRZ
1 NRZI
bitrt 50 – 64000

status Gets the current state of link under user control.
int status( )
Returns: 0 – 9 Current state (see link_state table on previous page)

trans Transmits a frame.
int trans (frame,address,len)
int frame, len;
char *address;
frame selects type of frame to transmit:
0x80 I–frame Sequenced I–frame
0x81 UI Unnumbered I–frame (NSI)
0x82 XIDC XID command frame
0x83 XIDR XID response frame
address is a pointer to the first byte of the message. len is the length of the message to be transmitted.
Returns: 0 Successful
int transmit (xloc, xlen)
char *xloc;
int xlen;
xloc is a pointer to the first byte of the message. xlen is the length of the message to be transmitted.
Returns: 0 Successful

trui Transmit a message in an unnumbered I–frame (UI frame).
int trui (xloc, xlen)
char *xloc;
int xlen;
xloc is a pointer to the first byte of the message. xlen is the length of the message to be transmitted.
Returns: 0 Successful

trxcni Transmits an XID command frame with no data field.
int trxcni ( )
Returns: 0 Successful

trxidc Transmits a message in an XID command frame.
int trxidc (xloc, xlen)
char *xloc;
int xlen;
xloc is a pointer to the first byte of the message. xlen is the length of the message to be transmitted.

trxidr Transmit a message in an XID response frame.
int trxidr (xloc, xlen)
char *xloc;
xloc is a pointer to the first byte of the message. xlen is the length of the message to be transmitted.
Returns: 0 Successful

trxrni Transmits an XID response frame with no data field.
int trxrni ( )
Returns: 0 Successful

V.120 LIBRARY

V.120 LIBRARY Filename: libv120.a

GET_FREELINK() Gets the number of first disabled link.
int get_freelink()
Returns:

0 – 63	Disabled link number
–1	No free links
–2	initp1 not performed

GET_FWAITING Gets the number of I–frames waiting to be transmitted on link.
int get_fwaiting (lnkn)
char lnkn; 0 – 63
Returns: 0 – 7 No. of I–frames

GET_LINK() Gets the number of the link currently under user control.
int get_link( )
Returns:
0 – 63 Current link number

GET_LLI() Gets the LLI of the link currently under user control.
int get_lli()
Returns:
0 – 63 Current link number

GET_LNKLLI Gets the LLI value for link lnkn.
int get_lnklli (lnkn)
char lnkn; 0 – 63
Returns

0 – 0x1fff	LLI value
>0x1fff	Link lnkn is disabled

GET_MESWAITING Gets no. of messages waiting to be received from the FEP.
int get_meswaiting ( )
Returns:
0 – 32 No. of messages

GET_RLINK() Gets the number of the link which sent the last received message.
int get_rlink( )
Returns: 0 – 63 Current link

–1	No messages recd
–2	initp1 not performed

GET_RXSTAT() Gets the low order byte of the frame status byte **frstat** for the last received message.
int get_rxstat()
Returns:

0–0xC3	frstat value
0xFF	No messages recd
0xFE	initp1 not performed

GET_SCONFIG ( ) Returns a copy of the current control configuration byte.
int get_sconfig ( )

GET_WINDOW Gets the number of outstanding I–frames on link number **lnkn.**
int get_window (lnkn)
char lnkn; 0 – 63
Returns: 0 – 7 No. of I–frames

INITP1 Starts the simulator (same as start_sim).
int initp1 (interface, sta, encode, bitrt)
int interface, sta, encode;
long bitrt;

interface	0	V–type interface (DCE)
	1	V–type interface (DTE)
	2	ISDN interface
sta	0	NETWORK
	1	SUBSCRIBER
encode	0	NRZ
	1	NRZI
bitrt	50 – 64000	

LINK_STAT Gets the current state of link n.
int link_stat(n)
char n; 0 – 63
Returns: 0 – 9 Current state

0	Link Disconnected
1	Link Connection Requested
2	Frame Rejected
3	Disconnect Requested
4	Information Transfer
5	Local Station Busy
6	Remote Station Busy
7	Local & Remote Station Busy
8	Remote Stn not Responding
9	Link Disabled

RECEIVE Receives a message from the FEP
int receive(dest_addr)
char *dest_addr;

S_N200 Sets the maximum number of retries (N200).
int s_n200 (val)
int val; 1 – 255
Returns: 0 Successful

S_N201 Sets the maximum length for an I–frame (N201).
int s_n201 (val)
int val; 1 – 512
Returns: 0 Successful

S_T200 Sets the time allowed for the remote station to respond (T200). Setting this value to 0 disables the T200 timer.
int s_t200 (val)
int val; 0 – 255
Returns: 0 Successful

S_T203 Sets the maximum time between frames (T203). Setting this value to 0 disables the T203 timer.
int s_t203 (val)
int val; 0 – 255
Returns: 0 Successful

SET_SCONFIG Sets the value of the control configuration byte
int set_sconfig (byte)
int byte;
Returns: 0 Successful

SET_LINK Puts link n under user control.
int set_link(n)
char n; 0 – 63
Returns:

0	Successful
–1	Parameter out of range
–2	initp1 not performed
–3	Timeout

SET_LLI Sets the LLI value for the link under user control. A value over 0x1FFF disables the link.
int set_lli(val)
int val; 0x00 – 0xFFFF hex
Returns:

0	Successful
–1	Parameter out of range
–2	initp1 not performed
–3	Timeout

SET_WINDOW Sets the maximum number of outstanding frames on each link.
int set_window (val)
int val; 1 – 7
Returns: 0 Successful

SETFLG Selects an interframe fill pattern.
int setflg (flag)

int flag;	0	0x7E fill
	1	0xFF fill
Returns:	0	Successful

SLOF ( ) Sends a DISC and waits for a UA.
int slof ( )
Returns: 0 Successful

SLON ( ) Sends a SABME and waits for a UA.
int slon ( )
Returns: 0 Successful

SRCH_LNK Returns the number of lowest link matching the specified SAPI/TEI.
int srch_lnk(sapi,tei)

Returns:	0 – 63	Link no.
	–1	No match

STATUS( ) Gets the current state of current link.
int status()
Returns: 0 – 9

START_SIM Starts the simulator (same as initp1).
n=start_sim (interface, sta, encode, bitrt)
int n, interface, sta, encode;
long bitrt;

interface	0	V–type (DCE)
	1	V–type (DTE)
	2	ISDN interface
sta	0	NETWORK
	1	SUBSCRIBER
encode	0	NRZ
	1	NRZI
bitrt	50 – 64000	

TRANS Transmits a frame.
int trans (frame,address,len)
frame selects type of frame to transmit:

0x80	I–frame	Sequenced I–frame
0x81	UI	Unnumbered I–frame (NSI)
0x82	XIDC	XID command frame
0x83	XIDR	XID response frame

address is a pointer to the first byte of the message. len is the length of the message.
Returns: 0 Successful

TRANSMIT Transmits a message in a sequenced (numbered) I–frame.
int transmit (xloc, xlen)
char *xloc;
int xlen;

TRANS_RESP Transmits a message in a sequencedI–frame response.
int trans_resp (xloc, xlen)
char *xloc;
int xlen;

TRUI Transmit a message in an unnumbered I–frame (UI frame).
int trui (xloc, xlen)
char *xloc;
int xlen;

TRXCNI Transmits an XID command frame with no data field.
int trxcni ( )

TRXIDC Transmits a message in an XID command frame.
int trxidc (xloc, xlen)
char *xloc;
int xlen;

TRXIDR Transmit a message in an XID response frame.
int trxidr (xloc, xlen)
char *xloc;

TRXRNI Transmits an XID response frame with no data field.
int trxrni ()

MULTI–LINK HDLC LIBRARY QUICK REFERENCE

<u>MULTI–LINK HDLC LIBRARY Filename:</u>

libmhdlc.a

flush Clears the receive buffer of the currently selected port.
flush ()
Returns: None

flush_all Clears the reception buffer of both ports.
flush_all()
Returns: None

init_a Initializes Port A.
int init_a (interface, sta, encode, bitrt)
int interface, sta, encode;
long bitrt;

interface	*0*	*DCE*	
	1	DTE	
	2	ISDN	
sta	0	Network	
	1	Subscriber	
encode	0	NRZ	
	1	NRZI	
bitrt	50 to 64000 bps		
Returns:	0	Successful	
	–1	Parameter error	

init_b Initializes Port B.
int init_b (interface, sta, encode, bitrt)
int interface, sta, encode;
long bitrt;

interface	*0*	*DCE*
	1	DTE
	2	ISDN
sta	0	Network
	1	Subscriber
encode	0	NRZ
	1	NRZI
bitrt	50 to 64000 bps	
Returns:	0	Successful
	–1	Parameter error

initp1 Initializes Ports A and B.
int initp1 (interface, sta, encode, bitrt)
int interface, sta, encode;
long bitrt;

interface	*0*	*DCE*
	1	DTE
	2	ISDN
sta	0	Network
	1	Subscriber
encode	0	NRZ
	1	NRZI
bitrt	50 to 64000 bps	
Returns:	0	Successful
	–1	Parameter error

mlh_flush Clears the receive buffer of the specified port.
mlh_flush (port)
int port;

port	0	Port A
	1	Port B

mlh_receive Causes the Chameleon to check for a received packet.
int mlh_receive (loc)
char *loc;
loc Pointer to the receive buffer.
It sets the global variable **rec_port**, as follows:

0	No packet was received
1	Packet received from Port A
2	Packet received from Port B

Returns:

0	No packet in buffer
2	FEP not initialized
128	Packet received

mlh_set_n1 Sets the N1 value for the specified port.
int mlh_set_n1 (port,val)
int port,val;

port	0	Port A
	1	Port B
val	N1 value (1 to 512)	
Returns:	0	Successful
	–1	Parameter error

mlh_set_n2 Sets the N2 value for the specified port.
int mlh_set_n2 (port,val)
int port,val;

port	0	Port A
	1	Port B
val	N2 value (1 to 512)	
Returns:	0	Successful
	–1	Parameter error

mlh_set_net Sets the specified port to act as a network.
int mlh_set_net (port)
int port,val;

port	0	Port A
	1	Port B
Returns:	0	Successful

mlh_set_t1 Sets the value of the T1 timer for the specified port.
int mlh_set_t1 (port,val)
int port,val;

port	0	Port A
	1	Port B
val	T1 value (1 to 255)	
Returns:	0	Successful
	–1	Parameter error

MULTI–LINK HDLC LIBRARY QUICK REFERENCE (CONTINUED)

mlh_set_t2 Sets the value of the T2 timer for the specified port.
int mlh_set_t2 (port,val)
int port,val;
port 0 Port A
1 Port B
val T2 value (1 to 255)
Returns: 0 Successful
–1 Parameter error

mlh_slof Disconnects the link on the specified port.
int mlh_slof (port)
int port;
port 0 Port A
1 Port B

mlh_slon Attempts to establish a link on the specified port.
int mlh_slon (port)
int port;
port 0 Port A
1 Port B

mlh_status Returns the link status of the specified port.
int mlh_status (port)
int port;
port 0 Port A
1 Port B
Returns:
0 Disconnected
1 Link conn. requested
2 Frame reject state
3 Link disconn. req.
4 Information xfer state
5 Local station busy
6 Remote station busy
7 Local & remote station busy

mlh_set_sub Sets the specified port to act as a subscriber.
int mlh_set_sub (port)
int port;
port 0 Port A
1 Port B
Returns: 0 Successful

mlh_set_window Sets the window size for the specified port.
int mlh_set_window (port,val)
int port,val;
port 0 Port A
1 Port B
val Window size (1–7)
Returns: 0 Successful
–1 Parameter error

mlh_trans Transmits a data packet on Port A or B as determined by the distribution pattern set by a call to the *set_pat* or the *set_ratio* function.
int mlh_trans (xloc,xlen)
char *xloc;
int xlen;
xloc Pointer to the packet
xlen Length of the packet
Returns: 0 Successful

receive Checks the reception buffer of the specified port for a received packet.
int receive (port,loc)
char *loc;
int port;
port 0 Receive from Port A
1 Receive from Port B
loc A pointer to receive buffer.
Returns:
0 No packet in buffer
2 FEP not initialized
128 Packet received

set_n1 Sets the N1 value for both ports.
int set_n1 (val)
int val;
val N1 value (1 to 512)
Returns: 0 Successful
–1 Parameter error

set_n2 Sets the N2 value for both ports.
int set_n2 (val)
int val;
val N2 value (1 to 512)
Returns: 0 Successful
–1 Parameter error

set_net Configures both ports to act as networks.
int set_net
Returns: 0 Successful

set_pat Specifies a user defined distribution pattern for transmitting packets using mlh_trans().
int set_pat (pat_ptr)
char *pat_ptr;
pat_ptr A pointer to a user defined table
The distribution pattern is defined in a table which contains the following values:
0 End of table
1 Send on Port A
2 Send on Port B

MULTI–LINK HDLC LIBRARY QUICK REFERENCE (CONTINUED)

set_ratio Selects a distribution pattern for transmitting packets using mlh_trans(). It specifies the percentage of packets to be transmitted over Port A.

int set_ratio (pct_a)
int pct_a;

pct_a The percentage of packets to be transmitted over Port A. Valid values are 0 to 100 in increments of 10, and –1. For example:

–1 All packets are transmitted over both Ports A and B.
0 0% of the packets are transmitted over Port A.
10 10% of the packets are transmitted over Port A.

Returns: 0 Successful
–1 Parameter error

set_sub Configures both ports to act as subscribers.

int set_sub ()

Returns: 0 Successful

set_t1 Sets the T1 timer to an identical value for both ports.

int set_t1 (val)
int val;

val T1 value (1 to 255)

Returns: 0 Successful
–1 Parameter error

set_t2 Sets the T2 timer to an identical value for both ports.

int set_t2 (val)
int val;

val T2 value (1 to 255)

Returns: 0 Successful
–1 Parameter error

set_window Sets the window size to an identical value for both ports.

int set_window (val)
int val;

val Window size (1 to 7)

Returns: 0 Successful
–1 Parameter error

slof Disconnects the link on both ports .

int slof ()

slon Attempts to establish a link on both ports by sending a SABM.

int slon ()

status Returns the link status of the currently selected port.

int status ()

Returns:
0 Disconnected
1 Link conn. requested
2 Frame reject state
3 Link disconn. req.
4 Information xfer state
5 Local station busy
6 Remote station busy
7 Local & remote stations busy

transmit Transmits a packet over the specified port.

int transmit (port,xloc,xlen)
char port,*xloc;
int xlen;

port 0 Port A
1 Port B
xloc Pointer to the packet
xlen Length of the packet

Returns: 0 Successful

Application Programmer's Interface C LIBRARY: libui.a

The Application Programmer's Interface C Library provides function which enable you to develop applications with pull–down menu interfaces. The library contains the functions and commands described below. For descriptions of the data structures used by the functions, refer to the Application Programmer's Interface manual.

addNewLine Inserts one line at a time to a list selector.

```
addNewLine (s, str)
SCRAREA     *s;
byte        *str;
```

*s Pointer to scrolling area of a BOX-REQ.

*str Pointer to the string to be inserted.

Returns: None

box_input BOX_INPUT creates a list box of selections at run–time. See box_req for more information.

box_req The structure type BOXREQ is used to define the box.

cSToggle Marks a specified position within a box or list selector with a character.

```
cSToggle (s, n, mode, ch, ch1)
SCRAREA     *s;
int         n, mode;
char        ch, ch1;
```

s	A pointer to the scrolling area within a BOXREQ
n	Position within the box
mode	0 Toggle 1 Set
ch	First marker character
ch1	Second marker character
Returns:	0 Set to first marker, ch. 1 Set to second marker, ch1

dsp_req Displays text within a window. The structure type DSPREQ is used to specify the information to be displayed.

erase_field Request to erase a specific field from a window. This will erase both the description and the associated value.

eraseb_req This request that an entire list box be erased from the screen. The structure required for this request is of the type ERASEREQ.

erasew_req This requests that a window be erased from the screen. The structure required for this request is of the type ERASERQ.

eraseEOS This function erases the screen from line 3 downward. It is useful in conjunction with pull down menu logic.

```
eraseEOS( )
```

Returns: None

fillBoxArea This function initializes the scrolling linked list located within the structure BOXREQ. This must be done once, typically in the beginning of the program, before a box or list selector can be accessed through a call to userInterface().

```
fillBoxArea (req, strlist)
BOXREQ      *req;
byte        *strlist[];
```

req	A pointer to BOXREQ
strlist	Address of the array containing the strings to be entered in the list box
Returns:	None

getBoxArea This function allocates space to the scrolling area of a list selector. The linked list is also initialized. If the area needs to be re–initialized at any point, this function can be called again.

```
getBoxArea (breq)
BOXREQ      *breq;
```

Returns: None

getFileChoice This function displays a list of files. The function reads the directory specified by the path for each occurence of a file with the specified extension. For each occurrence, the filename is loaded into the list selector.

```
getFileChoice (boxName, fPath, ext, bTitle, errMsg, insFlag, inserts, fnum, conf)
BOXREQ      *boxName;
byte        *fPath;
byte        *ext;
byte        *bTitle;
byte        *errMsg;
int         insFlag
byte        ** inserts;
int         fnum;
BOXCONF     *conf;
```

*boxName	A pointer to BOXREQ
*fpath	Directory path
*ext	File extension
*bTitle	Title string to be displayed
*errMsg	Error string that will be displayed at if no files exist
insFlag	When set to TRUE, this inserts the number of lines specified in fnum into the list box. Otherwise, set to FALSE.
**inserts	A pointer to an array of strings to be inserted when insFlag = TRUE. Otherwise set to NIL.
fnum	When insFlag is set to true, this is the number of lines to be inserted.
*conf	A pointer to the BOXCONF structure.
Returns:	The BOXCONF structure contains exit information.

initUI This function initializes the user interface. initUI() must be called before any other call is made to the interface.

initUI (dsp, box, req, nw, nb)

```
DISPLAY   *dsp;
BOX       *box;
WINDOWREQ *req;
int       nw;
int       nb;
```

*dsp	A pointer to the window administration area
*box	A pointer to the list box administration area
*req	window initiation of ERROR_WINDOW (This is required)
nw	NUM_OF_WINDOWS
nb	NUM_OF_BOXES
Returns:	None.

input_req This command displays a sequence of fields to be edited. The following keys can be used during runtime operation to modify the field values.

CTRL–N	Go to the next field
CTRL–P	Go to the previous field
CTRL–I	Insert mode (Default=overwrite)
CTRL–D	Delete to end of line
CTRL–A	Go to the beginning of the line
CTRL–E	Go to the end of the line
RETURN	Go to the next field
Space Bar	Toggle between preset values

There are three types of structures required to initiate an INPUT_REQ.

The INPREQ structure defines the location and color of parameters displayed, the prompt text and other messages.

The INPUT_FIELD_TYPE structure defines a field on the screen. To define a sequence of fields, an array of these structures is declared. The last entry of this array is defined as {0, 0, 0, 0 ,0 , ...} or zero for all values.

The FKEY_FIELD_TYPE structure defines the preset acceptable values for a field.

rel_req This request is used to de–allocate the memory set aside for a window and releases the associated window number. This should be done when a window will not be used again.

unMark This function removes all marks used to identify selections within a list box.

unMark (s)

```
SCRAREA *s;
```

*s	A scroll area within the box to be cleared
Returns:	None

userInterface This function gives the user access to the user interface. Each library request or command is initiated through a call to this function.

userInterface (req, conf, dsp, box)

```
byte     *req;
byte     *conf;
DISPLAY  *dsp;
BOX      *box:
```

req	A pointer to the structure containing the request type or event
box	A pointer to the list box administration area

The output is put in a structure of the type CONFIRM, where applicable.

window_req The command WINDOW_REQ can be used to initialize a window which will display information or it can display a frame around an input request.

The parameters for the WINDOW_REQ are incorporated into four structures:

WINDOWREQ

FIELD

FIELD_DEF

FIELD_SEQ

TASK COMMUNICATION C LIBRARY

TASK COMMUNICATION C LIBRARY: libcom.a

MB_MESS structure:

```
   typedef struct { int  port ;
                    int  type ;
                    int  info ;
                    int  len ;
                    char *pdata ;
      }             MB_MESS ;
```

port — Origin or the destination of a message: PORT_A or PORT_B

type — Type of message sent or received:

CT_ERR	error message
CT_DATA	data message
CT_EXIT	exit message
CT_CMD	command message
CT_FLUSH	flush the reception buffer

info — User defined field when type= CT_ERR or type = CT_CMD

len — Length of data if type=CM_DATA

pdata — Pointer to the data received or sent.

The library provides the following functions:

com_chkmb Checks for received messages.

```
#include "com.h"
int com_chkmb(pmess)
MB_MESS  *pmess ;
```

pmess pointer to a MB_MESS structure describing the received message.

Within this structure, these items are required for all received messages:

port For control tasks, indicates the port from which the message originated as: PORT_A or PORT_B

type Type of message received:

CT_ERR
0CT_CMD
0CT_DATA
CT_FLUSH
CT_EXIT

info User defined information if type =CT_ERR or type = CT_CMD

len Length of the data if type = CT_DATA.

pdata Pointer to the data received if type = CT_DATA.

Returns: 0 Message received
−1 No message received

com_crctlmb Called by the control task to create the communication channels for the control task.

```
#include "com.h"
        int  com_crctlmb(np)
        int  np ;
```

np Number of channels to the control task (1 or 2)

Returns: 0 Successful
−1 Insufficient sys. resources

com_crpmb Called by the protocol tasks to create the communication channels for the protocol tasks.

```
#include "com.h"
        int com_crpmb(port,np)
int  port ;
int  np ;
```

port Port where protocol task is going to run: PORT_A or PORT_B

np Number of channels to the control task (1 or 2)

Returns: 0 Successful
−1 Insufficient sys. resources

com_dlctlmb Closes communication channels. It must be called by the control task prior to terminating.

```
#include "com.h"
        void com_dlctlmb()
```

Returns: None

com_error Reports an error to the control task and is called by a protocol task.

```
#include "com.h"
        void com_error(port,status)
int  port ;
int  status ;
```

port Port where protocol task will run: PORT_A or PORT_B

status User field that can be used to tell more about the error to the control task.

Returns: None

com_exit Sends an EXIT message to a protocol task. It is called by the control task.

```
#include "com.h"
        void  com_exit(port,lid)
int  port ;
long lid ;
```

port Port where protocol task to kill is running: PORT_A or PORT_B

lid Loader ID returned by the function com_startl

Returns: None

com_flush Flushes the reception channel and then sends a flush message to the protocol task.
#include "com.h"
void com_flush(port)
int port ;
port Port where the protocol task is running: PORT_A or PORT_B
Returns: None

com_gptr Gets a pointer to the area containing data to transmit
#include "com.h"
char *com_gptr(size)
int size ;
size Size of memory to allocate
Returns: (char *)0 (NULL pointer)
> 0L Value of pointer

com_rel Releases memory allocated for data messages.
#include "com.h"
void com_rel(pframe)
char *pframe ;
pframe Pointer to the data received
Returns: None

com_setrdy Informs the control task that a protocol task is ready to receive a message. This function must be called by a protocol task during its initialization.
#include "com.h"
void com_setrdy(port)
int port ;
port Port on which the protocol task is running: PORT_A or PORT_B
Returns: None

com_snd Sends a message to a destination task using a communication channel. This function is called by both the protocol tasks and the control task in order to communicate to each other.
#include "com.h"
void com_snd(pmess)
MB_MESS *pmess ;
pmess Pointer to the MB_MESS structure containing the description of the message
Within this structure, the following items are required:
port For the control task, this is the destination task port. For the protocol task port, this is the port of the origin of the message.
PORT_A
PORT_B
type Type of message to be sent

The remaining items of the structure are required depending on the type of message sent.
–For CT_ERR or CT_CMD, the field info needs to be filled.
–For CT_EXIT and CT_FLUSH, no additional fields are needed.
For CT_DATA, the following fields are required:
len Length of the data
pdata Pointer to the data
Returns: 0 Message sent correctly
–1 Queue is full

com_startl Loads and starts a protocol task. This function is called by the control task.
#include "com.h"
long
com_startl(name,arg0,arg1,...,0L)
char *name ;
char *arg0,*arg1,... ;
name pointer to name of file
arg0 pointer to name of program
arg1 pointer to first parameter
.
.
0L Ends list of parameters
Returns: < 0 Loader error
> 0 Loading and starting OK and loader ID value

com_wrdy Causes control task to wait for the protocol task to be ready.
#include "com.h"
#include "mtosux.h"
int com_wrdy(port,delay)
int port ;
long delay ;
port Port on which the protocol task is running: PORT_A or PORT_B
delay Maximum delay allowed by the control task,in the format: time unit + number of units
time units: MS, TMS, HMS SEC, MIN, HRS, DAY
number of units: 0 – 255
Example: 30 + SEC
Returns: 0 Already received from the protocol task
–1 Timeout

V.24 INTERFACE

The electrical characteristics of V.24 series plugs on the Chameleon conform to the CCITT V.28 Recommendation.

The V.24 series plugs have the following electrical specifications:

Line Receiver:

•	Impedance:	6<Z<8(Kohms)
•	Max. Input Voltage	± 25 V
•	Decision Threshold	± 3 V

Line Transmitter:

•	Impedance:	< 100 ohms
•	Output Voltage:	± 12 V

The connectors of the V.24 series are 25 pin socket connectors of the standard ISO DB 25.

V.24 PIN ASSIGNMENTS
Monitoring Mode

DB25 Pin No.	CCITT Circuit No.	EIA	Ground	Incoming	Out-going	Processed by Chameleon	RS232 NAME
1	101	AA	x	x		x	Frame Ground
2	103	BA		x		x	Transmitted Data
3	104	BB		x		x	Received Data
4	105	CA		x		x	Request to Send
5	106	CB		x		x	Clear to Send
6	107	CC		x		x	Data set Ready
7	108	AB	x	x		x	Signal Ground
8	109	CF		x		x	Data Carrier Detect
9							+ dc Test Voltage
10							– dc Test Voltage
11							Unassigned
12	122	SCF		x		x	2nd Data Carrier Detect
13	121	SCB					2nd Clear to Send
14	118	SBA					2nd Transmitted Data
15	114	DB		x		x	Transmitted Clock
16	119	SBB					2nd Received Data
17	115	DD		x		x	Receiver Clock
18							Receiver Dibit Clock
19	120	SCA					2nd Request to Send
20	108.2	CD		x		x	Data Terminal Ready
21	110	CG					Signal Quality Detect
22	125	CE		x		x	Ring Indicator
23							Data Rate Select
24	113	DA		x		x	Ext. Transmitter Clock
25							Busy

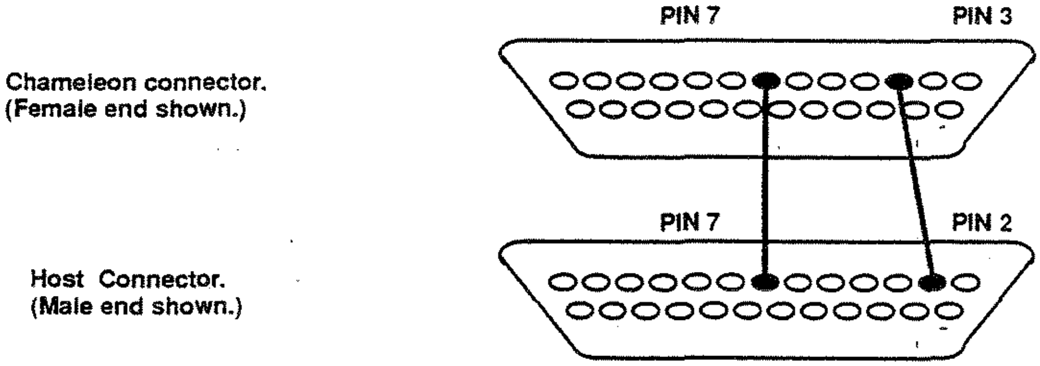
V.24 PIN ASSIGNMENTS
Simulation Mode

DB25 Pin No.	CCITT Circuit No.	EIA	Ground	To DCE	From DCE	Processed by Chameleon	RS232 NAME
1	101	AA	x			x	Frame Ground
2	103	BA		x		x	Transmitted Data
3	104	BB			x	x	Received Data
4	105	CA		x		x	Request to Send
5	106	CB			x	x	Clear to Send
6	107	CC			x	x	Data set Ready
7	108	AB	x			x	Signal Ground
8	109	CF			x	x	Data Carrier Detect
9							+ dc Test Voltage
10							– dc Test Voltage
11							Unassigned
12	122	SCF			x	x	2nd Data Carrier Detect
13	121	SCB			x		2nd Clear to Send
14	118	SBA		x			2nd Transmitted Data
15	114	DB			x	x	Transmitted Clock
16	119	SBB			x		2nd Received Data
17	115	DD			x	x	Receiver Clock
18							Receiver Dibit Clock
19	120	SCA		x			2nd Request to Send
20	108.2	CD		x		x	Data Terminal Ready
21	110	CG			x		Signal Quality Detect
22	125	CE			x	x	Ring Indicator
23							Data Rate Select
24	113	DA		x		x	Ext. Transmitter Clock
25							Busy

RS232 CABLE

The DTE must be provided with an extension cable no longer than fifty feet. Longer cables are permitted only if the load capacitance at the interface point does not exceed 2500 picofarads. Restricting cable connections to fifty feet between the computer communications adaptor and the local data set and between the remote data set and the associated terminal guards against excessive signal distortion.

For terminal emulation (page 10) and Kermit file transfer (page 11), you may have to use a special RS232 cable, depending on the device you are connecting to the Chameleon. This cable configuration requires pin 7 and 7 connected and pin 2 and 3 switched, as shown in the figure below.



V.35 INTERFACE

The V.35 interface module includes:

- One male connector (reference AMP 201 357–1)
- One female connector (reference AMP 200 838–2)
- Standard SAE 632 mounting hardware single lead jackscrew.

The male connector's male jackpost is near pin MM. The female connector's female jackscrew is near pin MM.

The diameter of the pins is 0.060" for units to be used in the U.S., Japan, Australia and England. For France, Switzerland and Sweden, the diameter is 0.040".

The pins can be removed or reassigned easily using an AMP tool (reference AMP 305 183).

Electrical Characteristics

The unbalanced signals have electrical characteristics which conform to the CCITT's V.28/EIA RS232.

Driver	Output voltage:	+/– 10 volts
	Output impedance:	300 ohms
	Output slew rate:	30 volts/microseconds
Receiver	Input resistance:	approximately 5 Kohms
	Input voltage max:	+/– 25 volts
	hysteresis:	3 to 4 volts

The balanced signals have electrical characteristics which conform to the CCITT's X.27/EIA RS422.

Driver	Output resistance:	200 ohms differential
	Lead to ground:	175 ohms
	Output current:	150 mA maximum
	Output voltage:	+/– 3 volts
Receiver	Input resistance:	200 ohms differential
	Lead to ground:	175 ohms
	Input sensitivity:	+/– 200 mvolts

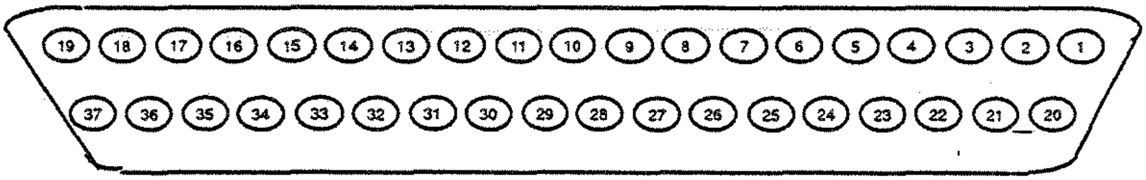
V.35 INTERFACE PIN ASSIGNMENT

Pin No.	CCITT Circuit No.	Name	From		Type		RS232 Name
			DCE	DTE	Bal	Unbal	
A		FG					Frame Ground
B	102	SG					Signal Ground
C	105	RTS		x		x	Request to Send
D	106	CTS	x			x	Clear to Send
E	107	DSR	x			x	Data Set Ready
F	109	DCD	x			x	Data Carrier Detect
H	108	DTR		x		x	Data Terminal Ready
J	125	RI	x			x	Ring Indicator
P	103	TD (A)		x	x		Transmit Data
R	104	RD (A)	x		x		Receive Data
S	103	TD (B)		x	x		Transmit Data
T	104	RD (B)	x		x		Receive Data
U	113	SCTE (A)		x	x		Transmitter Signal Timing
V	115	SCR (A)	x		x		Receiver Signal Timing
W	113	SCTE (B)		x	x		Transmitter Signal Timing
X	115	SCR (B)	x		x		Receiver Signal Timing
Y	114	SCT (A)	x		x		Transmitter Signal Timing
AA/a	114	SCT (B)	x		x		Transmitter Signal Timing

RS423/V.10/V.36 INTERFACE

The physical connection of interchange circuits within a data terminal and a data set is made by a pair of pluggable connectors (the interface point.) The Chameleon side is a 37 pin D–subminature socket (female) connector (DB37S).

The terminal side consists of the matching male connector (DB37P). The pinout below is shown as the connector is viewed from the rear of the machine:



Electrical Characteristics

This is an unbalanced signal which has electrical characteristics which conform to CCITT's V.10/EIA RS423.

Driver	Output voltage:	±
	Output impedance:	< 50 ohms
	Output current:	150 mA maximum
Receiver	Input Voltage:	± 10 volts
	Input impedance:	
	Input sensitivity:	± 200 mvolts

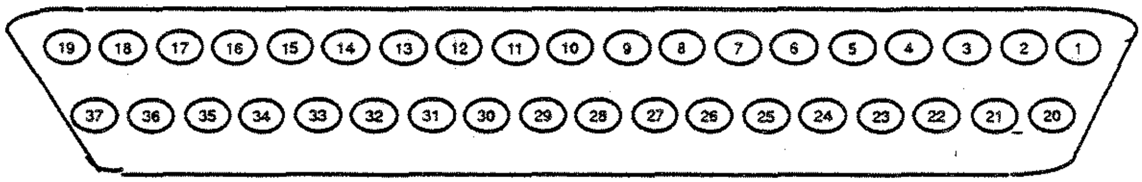
RS423/V.10/V.36 CONNECTOR PINOUT

DB37 Pin Number	ISO Circuit	CCITT Circuit Mnemonic and Name		Circuit Direction	Circuit Type		Implem-ented by Chameleon
19	102	SG	Signal ground	–			X
37	102a	SC	Send Common	To DCE	Common		X
20	102b	RC	Receive Common	From DCE			X
28	135	IS	Terminal in Service	To DCE			
15	125	IC	Incoming Call	From DCE	Control		
12 / 30	108	TR	Terminal Ready	To DCE			X
11 / 29	107	DM	Data Mode	From DCE			X
4 / 22	103	SD	Send Data	To DCE	Data	P	X
6 / 24	104	RD	Receive Data	From DCE		R	X
17 / 35	113	TT	Terminal Timing	To DCE		I	X
5 / 23	114	ST	Send Timing	From DCE	Timing	M	X
8 / 26	115	RT	Receive Timing	From DCE		A	X
7 / 25	105	RS	Request to Send	To DCE		R	X
9 / 27	106	CS	Clear to Send	From DCE		Y	X
13 / 31	109	RR	Receiver Ready	From DCE		C	X
33	110	SQ	Signal Quality	From DCE	Control	H	
34	136	NS	New Signal	To DCE		A	
16	111 / 126	SF	Select Frequency	To DCE		N	
16	111 / 126	SR	Signaling Rate Selector	To DCE		N	
2	112	SI	Signaling Rate Indicator	From DCE		E	
						L	
10	141	LL	Local Loopback	To DCE			
14	140	RL	Remote Loopback	To DCE	Control		
18	142	TM	Test Mode	From DCE			
32	116	SS	Select Standby	To DCE	Control		
36	117	SB	Standby Indicator	From DCE			

RS422/V.11/V.36 INTERFACE

The physical connection of interchange circuits within a data terminal and a data set is made by a pair of pluggable connectors (the interface point.) The Chameleon side is a 37 pin D–subminature socket (female) connector (DB37S).

The terminal side consists of the matching male connector (DB37P). The pinout below is shown as the connector is viewed from the rear of the machine:



Electrical Characteristics

RS422 is a Balanced Voltage Digital Signal with electrical characteristics which conform to the CCITT's V.11/X.27.

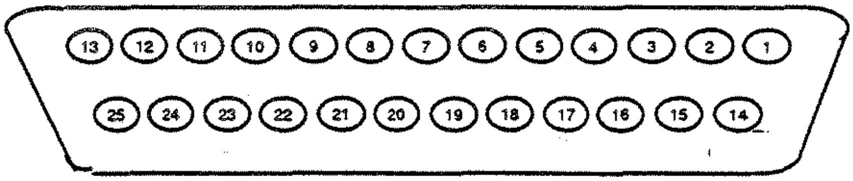
Driver	Output resistance:	200 ohms differential
	Lead to ground:	175 ohms
	Output current:	150 mA maximum
	Output voltage:	± 3 volts
Receiver	Input resistance:	200 ohms differential
	Lead to ground:	175 ohms
	Input sensitivity:	± 200 mvolts

RS422/V.11/V.36 CONNECTOR PINOUT

DB37 Pin Number	ISO Circuit	CCITT Circuit Mnemonic and Name		Circuit Direction	Circuit Type		Implem-ented by Chameleon
19	102	SG	Signal ground	–	Common		X
37	102a	SC	Send Common	To DCE			X
20	102b	RC	Receive Common	From DCE			X
28	135	IS	Terminal in Service	To DCE	Control		X
15	125	IC	Incoming Call	From DCE			X
12 / 30	108	TR	Terminal Ready	To DCE			X
11 / 29	107	DM	Data Mode	From DCE			X
4 / 22	103	SD	Send Data	To DCE	Data	PRIMARY CHANNEL	X
6 / 24	104	RD	Receive Data	From DCE			X
17 / 35	113	TT	Terminal Timing	To DCE	Timing		X
5 / 23	114	ST	Send Timing	From DCE			X
8 / 26	115	RT	Receive Timing	From DCE			X
7 / 25	105	RS	Request to Send	To DCE	Control		X
9 / 27	106	CS	Clear to Send	From DCE			X
13 / 31	109	RR	Receiver Ready	From DCE			X
33	110	SQ	Signal Quality	From DCE			
34	136	NS	New Signal	To DCE			
16	111 / 126	SF	Select Frequency	To DCE			
16	111 / 126	SR	Signaling Rate Selector	To DCE			
2	112	SI	Signaling Rate Indicator	From DCE			
10	141	LL	Local Loopback	To DCE	Control		
14	140	RL	Remote Loopback	To DCE			
18	142	TM	Test Mode	From DCE			
32	116	SS	Select Standby	To DCE	Control		
36	117	SB	Standby Indicator	From DCE			

PARALLEL PRINTER CONNECTOR PINOUT

The Chameleon parallel printer connector is a 25 pin D–sub socket (female) connector (DB25S). This connector is pinout and signal compatible with the IBM PC. It is also signal compatible with Centronics compatible parallel interface printers.The pinout is as shown below:

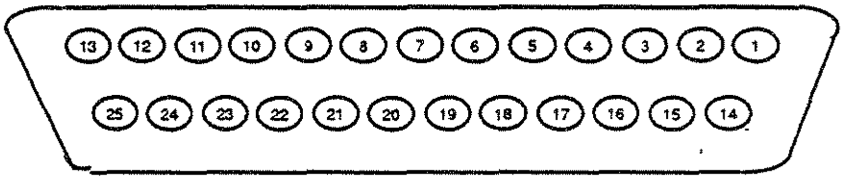


All signals are standard TTL levels.

Pin Number	Description
1	/STROBE (Active Low)
2	Data 0
3	Data 1
4	Data 2
5	Data 3
6	Data 4
7	Data 5
8	Data 6
9	Data 7
10	/ACK (Active Low)
11	Busy
12	No Connection
13	No Connection
14	No Connection
15	No Connection
16	No Connection
17	No Connection
18	Ground
19	Ground
20	Ground
21	Ground
22	Ground
23	Ground
24	No Connection
25	Ground

SERIAL PRINTER CONNECTOR PINOUT

The Chameleon serial printer connector is a 25 pin D–subminature socket (female) (DB25S). The pinout is shown as the connector is viewed from the rear of the machine:

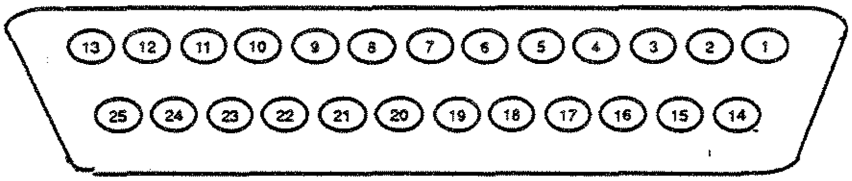


All signals are standard RS–232 voltage levels. The connector is physically and electrically a DCE type connector.

DB25 Pin No.	CCITT Circuit No.	EIA	Source	Signal Name	
1	101	AA	Chassis		Chassis Ground
2	103	BA	Printer	TXD	Transmit Data
3	104	BB	Chameleon	RXD	Receive Data
4	105	CA	Printer	RTS	Request to Send
5	106	CB	Chameleon	CTS	Clear to Send
6	107	CC	Chameleon	DSR	Data Set Ready
7	102	AB	Signal Gnd.	GND	Signal Ground
8	109	CF	Chameleon	DCD	Carrier Detect
15	114	DB	Chameleon	TXC	Transmit Clock
17	115	DD	Chameleon	RXC	Receive Clock
20	108	CD	Printer	DTR	Data Terminal Ready
24	–	DA	Printer	CK	External Clock

REMOTE I/O CONNECTOR PINOUT

The Chameleon Remote I/O connector is a 25 pin D–subminature socket (female) (DB25S). The pinout is shown as the connector is viewed from the rear of the machine:

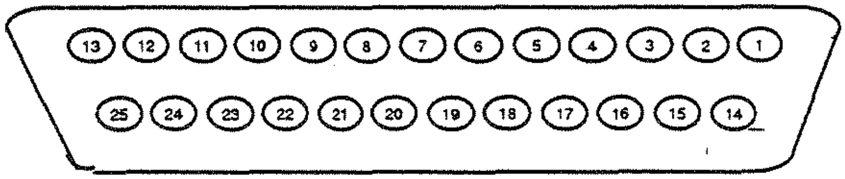


All signals are standard RS232 voltage levels. The connector is physically and electrically a DCE type connector.

DB25 Pin No.	CCITT Circuit No.	EIA	Source	Signal Name	
1	101	AA	Chassis		Chassis Ground
2	103	BA	Printer	TXD	Transmit Data
3	104	BB	Chameleon	RXD	Receive Data
4	105	CA	Printer	RTS	Request to Send
5	106	CB	Chameleon	CTS	Clear to Send
6	107	CC	Chameleon	DSR	Data Set Ready
7	102	AB	Signal Gnd.	GND	Signal Ground
8	109	CF	Chameleon	DCD	Carrier Detect
15	114	DB	Chameleon	TXC	Transmit Clock
17	115	DD	Chameleon	RXC	Receive Clock
20	108	CD	Printer	DTR	Data Terminal Ready
24	–	DA	Printer	CK	External Clock

AUX 1 AND AUX 2 PORTS CONNECTOR PINOUTS

The Chameleon Aux 1 and Aux 2 serial port connectors are 25 pin D–subminature sockets (female) (DB25S). The pinout for both is shown as the connector is viewed from the rear of the machine:

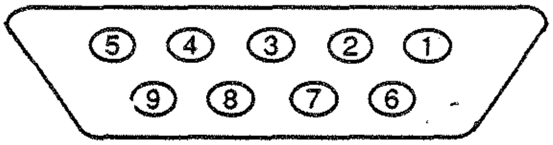


All signals are standard RS232 voltage levels. The connector is physically and electrically a DCE type connector.

DB25 Pin No.	CCITT Circuit No.	EIA	Source	Signal Name	
1	101	AA	N/C		Chassis Ground
2	103	BA	Terminal	TXD	Transmit Data
3	104	BB	Chameleon	RXD	Receive Data
4	105	CA	Terminal	RTS	Request to Send
5	106	CB	Chameleon	CTS	Clear to Send
6	107	CC	Chameleon	DSR	Data Set Ready
7	102	AB	Signal Gnd.	GND	Signal Ground
15	114	DB	Chameleon	TXC	Transmit Clock
17	115	DD	Chameleon	RXC	Receive Clock
20	108	CD	Terminal	DTR	Data Terminal Ready
22	125	CE	Terminal	RI	Ring Indicator
24	–	DA	Terminal	CK	External Clock

VIDEO CONNECTOR PINOUT

The Chameleon video connector is a 9 pin D–sub socket (female) connector (DB9S). The pinout is as shown below:



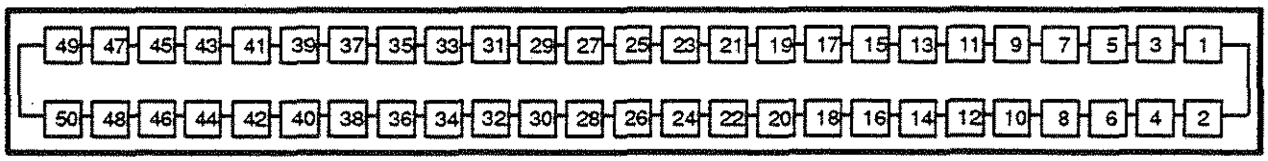
All signals are standard TTL levels.

Pin No.	Description
1	Ground
2	Ground
3	Red
4	Green
5	Blue
6	Intensity
7	Monochrome
8	Horizontal Sync.
9	Vertical Sync.

This connector is pinout and signal compatible with the IBM PC. The video signal requires a monitor capable of displaying 640 pixels by 240 lines (this is higher resolution than the standard PC CGA standard). High resolution or "Multisync" type monitors are recommended for use with the Chameleon.

SCSI INTERFACE SIGNALS

The Chameleon SCSI interface signals are as shown below. All signals are low true.

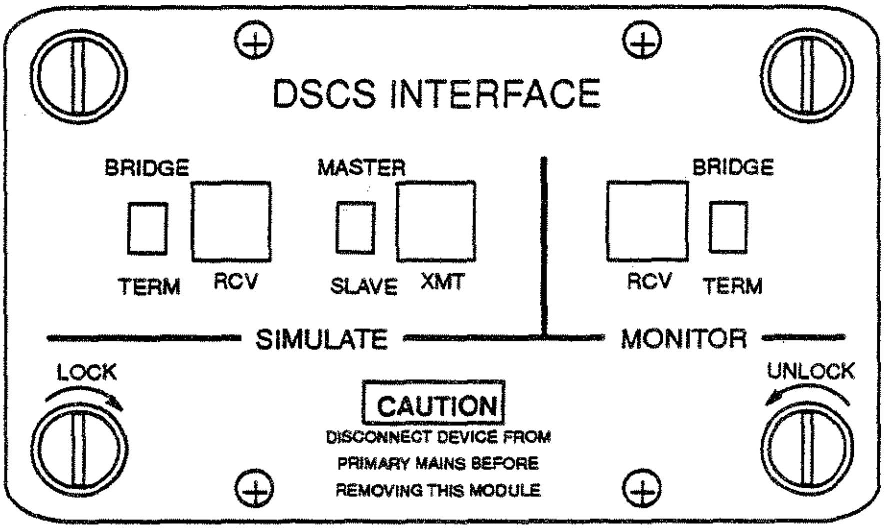


All odd pins are ground.

GND	1	2	Data Bit 0 (DB0).
.	3	4	Data Bit 1 (DB1).
.	5	6	Data Bit 2 (DB2).
.	7	8	Data Bit 3 (DB3).
.	9	10	Data Bit 4 (DB4).
.	11	12	Data Bit 5 (DB5).
.	13	14	Data Bit 6 (DB6).
.	15	16	Data Bit 7 (DB7).
.	17	18	Data Parity(DBP).
.	19	20	Open.
.	21	22	Open.
.	23	24	Open.
.	25	26	Open.
.	27	28	Open.
.	29	30	Open.
.	31	32	Open.
.	33	34	Open.
.	35	36	Busy (BSY).
.	37	38	Acknowledge (ACK).
.	39	40	Reset (RST).
.	41	42	Message (MSG).
.	43	44	Select (SEL).
.	45	46	Control/Data (C/D).
.	47	48	Request (REQ)GND
GND	49	50	Input/Output (I/O)

DSCS INTERFACE

The Digital Signal Customer Service (DSCS) interface has two receiver circuits and one transmitter circuit, enabling it to operate in either a Simulation or Monitoring mode. The figure below shows the DSCS Interface module as viewed from the rear of the machine.



The receiver (A) and transmitter (B) connections to the DSCS interface are industry–standard 3–conductor bantam jacks. The receivers operate with standard DSCS/DDS signals per AT&T Pub 62310 and Bellcore TA–TSY–000083. The maximum distance from OCU and DSU is 1000 feet.

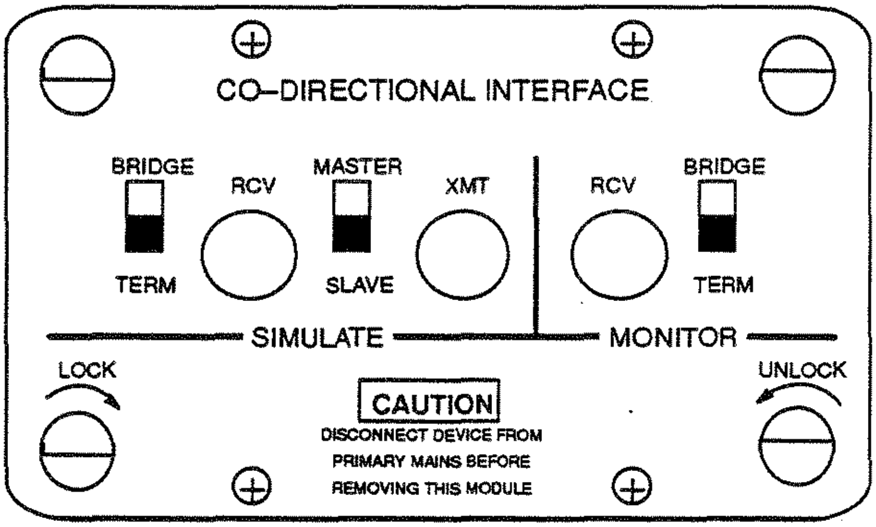
The transmitter provides a balanced output. The pulse amplitude and shape is in accordance with AT&T Pub 62310 and Bellcore TA–TSY–000083. The internal clock provides 56 KBPS 0.01%. This clock times the transmission when the Master/Slave switch is in the Master position. The pulse amplitude is 1.66 volts nominal. The encoding/decoding method is AMI with zero suppression.

SIMULATE
MODE In SIMULATE mode, the DSCS uses one transmitter (B) and one receiver (A). In this mode, the interface provides the clock to the Chameleon; therefore, the Chameleon must be configured as a DTE. The TERM/BRIDGE switch setting determines the input impedance, as follows:
TERM Terminated. Provides a 135 ohm nominal input impedance 5 ohms, balanced
BRIDGE Provides an input impedance greater than 3 K ohms, balanced
The Master/Slave switch selects the transmitter clock used by the DSCS Interface, as follows:
Master Transmits to the network using the internally generated clock
Slave Transmits to the network using the recovered received clock

MONITOR
MODE In MONITOR mode, the DSCS uses two receivers: the SIMULATE receiver (A) and the MONITOR receiver (A). A TERM/BRIDGE switch is provided for each receiver. For both receivers, the DSCS Interface derives a clock from the received signal for use in received timing.

G.703 CO–DIRECTIONAL INTERFACE MODULE

The CCITT G.703 Co–Directional Interface for the Chameleon 32 operates at 64 Kbps. It contains two receiver circuits and one transmitter circuit. This allows the interface to operate in either Simulation or Monitoring mode. The document used as a standard reference is the CCITT Red Book, Volume III – Fascicle III.3, Recommendation G.703. The figure below shows the Co–Directional Interface module as viewed from the rear of the machine.



In simulate mode, the Co–Directional interfaces uses both the transmitter and receiver. In this mode, the Co–Directional interface module must be configured as a DTE. The Master/Slave switch selects the transmitter clock source used by the Co–Directional interface, as follows:

•¢ When Master is selected, the transmit clock is generated by the internal clock of the Co–Directional interface.

•¢ When slave is selected, the transmit clock is derived from the recovered receive clock, and is thus synchronous to the receive clock.

In Monitor mode, the Co–Directional interfaces uses two receivers: the Simulate receiver and the Monitor receiver. Both receivers use the received clock for receive timing.

Each receiver is provided with a Term/Bridge switch. When Term is selected, the line is terminated with a 120 ohm nominal input impedance. When Bridge is selected, the input impedance is greater than 3k ohms. If multiple receivers are connected to one line, only one should be terminated, and the remaining receivers set for Bridge mode. If only one receiver is connected, it should be in Term mode.

Receivers Receivers operate with standard Co–Directional signals per CCITT Recommendation G.703.

- Coding method: per G.703
- Input Impedance:
  - 120 ohms 5 ohms, balanced (Term mode)
  - > 3k ohms, balanced (Bridge mode)
- Bipolar signal input range 5.0 Volts peak–to–peak to 0.3 Volts peak–to–peak

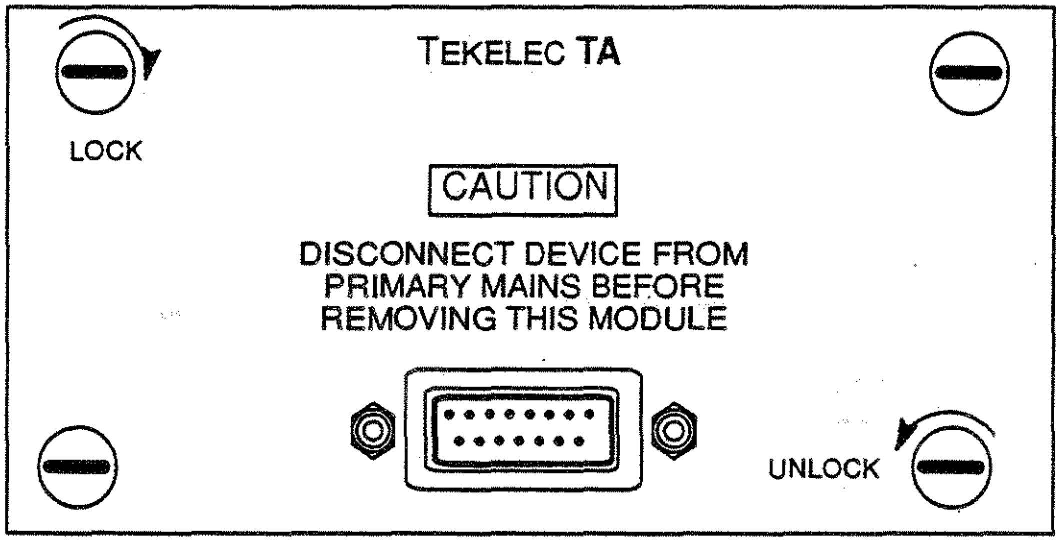
Transmitter The transmitter provides a balanced output.

- Output impedance: 120 ohms 5 ohms
- Pulse amplitude and shape is in accordance with CCITT Rec. G.703.
- Encoding method: per CCITT Rec. G.703
- Internal clock provides 64 KBPS 100 ppm.
- Pulse amplitude: 1.0 volts nominal, into 120 ohm balanced
- Peak voltage of no pulse: 0 0.1 volts

X.21 INTERFACE INTERFACE MODULE

The X.21 Interface is a combined hardware/software package that provides a physical interface for simulation and monitoring. The X.21 Interface module is shown in the figure below. The X.21 interface conforms to the following CCITT recommendations:

- CCITT recommendation X.21 1984
- CCITT recommendation V.11 1984
- CCITT recommendation X.4 1980



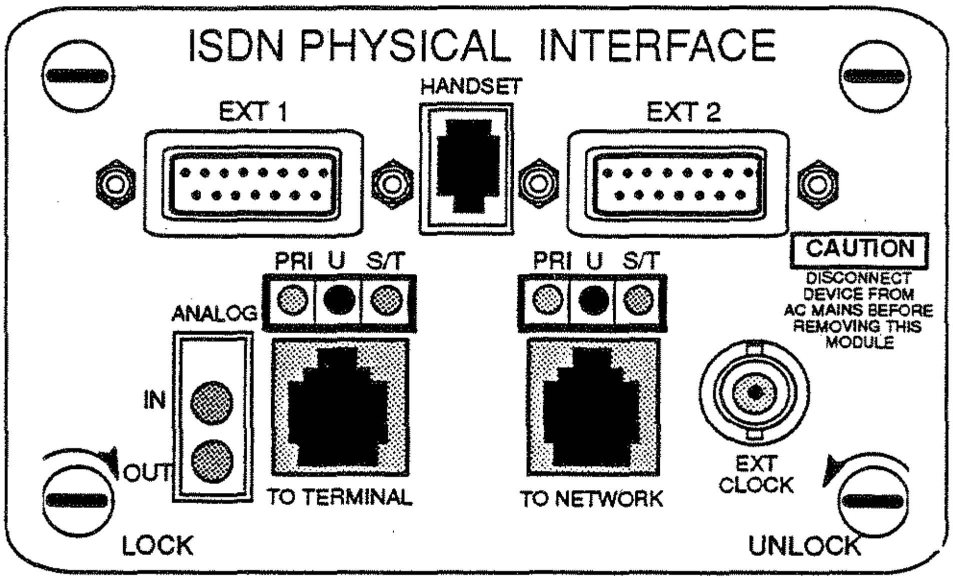
The 15 pin connector pin out is as follows:

Pin No.	Description
1	Shield
2	T(A) Transmit (A)
3	C(A) Control (A)
4	R(A) Receive (A)
5	I(A) Indication (A)
6	S(A) Signal Element Timing (A)
7	B(A) Byte Timing (A)
8	Ground
90	T(B) Transmit (B)
10	C(B) Control (B)
11	R(B) Receive (B)
12	I(B) Indication (B)
13	S(B) Signal Element Timing (B)
14	B(B) Byte Timing (B)
15	Reserved

Refer to the *Chameleon Protocol Interpretation Manual*, Chapter 18, for a description of the X.21 software.

U–INTERFACE I/O MODULE

The ISDN PHYSICAL INTERFACE is a combined hardware/software package for 2B1Q U–interface simulation and monitoring. Although designed to accommodate Basic Rate and Primary Rate, software is not presently available for these implementations. A more complete description of this hardware is found in Chapter 20: 2B1Q U–Interface of the *Protocol Interpretation Manual.*



The EXT1 and EXT2 connectors are 15–pin, D–subminiature females, DA15S type. The figures below give the pinouts for these bi–directional connectors. All signals are standard RS422 voltage levels.

An RS449 cable is provided with the ISDN 2B1Q U–INTERFACE package. The chart below correlates the pins of this cable with those of the DA15S connectors.

DA15S Pin Number	RS449 Pin Number	Description	Direction
1	1	Chassis Ground	
2	22	Send Data B	Input
3	–	Unused	–
4	24	Receive Data B	
5	–	Unused	–
6	23	Send Timing B	Output
8	19	Signal Ground	–
9	4	Send Data A	Inputs
10	–	Unused	–
11	6	Receive Data A	Output
12	–	Unused	–
13	5	Send Timing A	Output
14	–	Unused	–
15	8	Receive Timing A	Output