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.
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.

No part of this documentation may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying or recording, for any purpose without the express written permission of an authorized representative of Tekelec.

Copyright Tekelec 1991. All rights reserved.

Chameleon 32 is a registered trademarks of Tekelec.

Other product names used herein are for identification purposes only, and may be trademarks of their respective companies.

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

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIGURATION</td>
<td>1</td>
</tr>
<tr>
<td>MAIN CONFIGURATION MENU</td>
<td>2</td>
</tr>
<tr>
<td>APPLICATIONS SELECTION MENU</td>
<td>3</td>
</tr>
<tr>
<td>PAGE MANIPULATION KEYS</td>
<td>4</td>
</tr>
<tr>
<td>FILE FORMAT AND REQUIREMENTS</td>
<td>5</td>
</tr>
<tr>
<td>HARD DISK DIRECTORY STRUCTURE</td>
<td>6</td>
</tr>
<tr>
<td>FILE MANAGEMENT MENU</td>
<td>7</td>
</tr>
<tr>
<td>UTILITIES MENU</td>
<td>8</td>
</tr>
<tr>
<td>PRINT KEYS AND COMMANDS</td>
<td>9</td>
</tr>
<tr>
<td>REMOTELY CONTROLLING THE CHAMELEON</td>
<td>10</td>
</tr>
<tr>
<td>TERMINAL EMULATION</td>
<td>11</td>
</tr>
<tr>
<td>KERMIT FILE TRANSFER</td>
<td>12</td>
</tr>
<tr>
<td>BASIC RATE INTERFACE SETUP MENU</td>
<td>13</td>
</tr>
<tr>
<td>PRIMARY RATE INTERFACE SETUP MENU</td>
<td>15</td>
</tr>
<tr>
<td>2B1Q U-INTERFACE SETUP MENU</td>
<td>16</td>
</tr>
<tr>
<td>2B1Q SIMULATION CONFIGURATION</td>
<td>18</td>
</tr>
<tr>
<td>ANALYSIS CONTROL/SHIFT KEYS</td>
<td>21</td>
</tr>
<tr>
<td>HISTORY DISPLAY KEYS</td>
<td>22</td>
</tr>
<tr>
<td>DUAL LINE APPLICATION</td>
<td>23</td>
</tr>
<tr>
<td>BERT APPLICATION</td>
<td>24</td>
</tr>
<tr>
<td>DIRECT-TO-DISK APPLICATION</td>
<td>27</td>
</tr>
<tr>
<td>STATISTICS</td>
<td>29</td>
</tr>
<tr>
<td>STATISTICS</td>
<td>30</td>
</tr>
<tr>
<td>TRIGGERING APPLICATION</td>
<td>31</td>
</tr>
<tr>
<td>TRIGGERING CONDITIONS (FUNCTION KEYS)</td>
<td>32</td>
</tr>
<tr>
<td>SS#7 Level 3 and Level 4 Triggering Options</td>
<td>33</td>
</tr>
<tr>
<td>SIMULATOR ROAD MAP</td>
<td>34</td>
</tr>
<tr>
<td>FRAMEM LAPD DEFAULT MNEMONIC TABLE</td>
<td>35</td>
</tr>
<tr>
<td>FRAMEM HDLC/SDLC MNEMONIC TABLE</td>
<td>36</td>
</tr>
<tr>
<td>SIMP/L LAPD DEFAULT MNEMONIC TABLE</td>
<td>37</td>
</tr>
<tr>
<td>SIMP/L HDLC DEFAULT MNEMONIC TABLE</td>
<td>37</td>
</tr>
<tr>
<td>BSC DEFAULT MNEMONIC TABLE</td>
<td>38</td>
</tr>
<tr>
<td>ASYNC DEFAULT MNEMONIC TABLE</td>
<td>39</td>
</tr>
<tr>
<td>BASIC COMMANDS</td>
<td>40</td>
</tr>
<tr>
<td>FRAMEM COMMANDS</td>
<td>44</td>
</tr>
<tr>
<td>FRAMEM DMI COMMANDS</td>
<td>46</td>
</tr>
<tr>
<td>SIMP/L COMMANDS</td>
<td>47</td>
</tr>
<tr>
<td>ASYNC BASIC COMMANDS</td>
<td>50</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSC BASIC COMMANDS</td>
<td>51</td>
</tr>
<tr>
<td>SITREX COMMANDS</td>
<td>52</td>
</tr>
<tr>
<td>SITREX AUTOMATIC SIMULATOR COMMANDS</td>
<td>56</td>
</tr>
<tr>
<td>SITREX DEFAULT PSEUDO–USERS</td>
<td>56</td>
</tr>
<tr>
<td>SITREX TRACE PAGE COMMANDS</td>
<td>57</td>
</tr>
<tr>
<td>SITREX TRACE INTERPRETATION</td>
<td>58</td>
</tr>
<tr>
<td>SITREX ERROR CODES</td>
<td>59</td>
</tr>
<tr>
<td>C SHELL COMMANDS</td>
<td>60</td>
</tr>
<tr>
<td>C SHELL REFERENCE</td>
<td>61</td>
</tr>
<tr>
<td>C LIBRARY FUNCTIONS</td>
<td>62</td>
</tr>
<tr>
<td>vi COMMANDS</td>
<td>67</td>
</tr>
<tr>
<td>C PROTOCOL LIBRARY COMMON FUNCTIONS</td>
<td>69</td>
</tr>
<tr>
<td>AUTO HDLC C LIBRARY FUNCTIONS</td>
<td>70</td>
</tr>
<tr>
<td>BOP C LIBRARY FUNCTIONS</td>
<td>71</td>
</tr>
<tr>
<td>LAPD C LIBRARY FUNCTIONS</td>
<td>72</td>
</tr>
<tr>
<td>SDLC C LIBRARY FUNCTIONS</td>
<td>75</td>
</tr>
<tr>
<td>BASIC RATE INTERFACE LIBRARY FUNCTIONS</td>
<td>77</td>
</tr>
<tr>
<td>2B1Q U–INTERFACE C LIBRARY FUNCTIONS</td>
<td>79</td>
</tr>
<tr>
<td>BSC C LIBRARY FUNCTIONS</td>
<td>82</td>
</tr>
<tr>
<td>C PRIMARY RATE INTERFACE LIBRARY FUNCTIONS</td>
<td>83</td>
</tr>
<tr>
<td>ASYNC LIBRARY QUICK REFERENCE</td>
<td>85</td>
</tr>
<tr>
<td>C ANALYSIS LIBRARY FUNCTIONS</td>
<td>86</td>
</tr>
<tr>
<td>RS232 CABLE</td>
<td>103</td>
</tr>
<tr>
<td>V.35 INTERFACE</td>
<td>104</td>
</tr>
<tr>
<td>RS423/V.10/V.36 INTERFACE</td>
<td>106</td>
</tr>
<tr>
<td>RS423/V.10/V.36 CONNECTOR PINOUT</td>
<td>107</td>
</tr>
<tr>
<td>RS422/V.11/V.36 INTERFACE</td>
<td>108</td>
</tr>
<tr>
<td>RS422/V.11/V.36 CONNECTOR PINOUT</td>
<td>109</td>
</tr>
<tr>
<td>PARALLEL PRINTER CONNECTOR PINOUT</td>
<td>110</td>
</tr>
<tr>
<td>SERIAL PRINTER CONNECTOR PINOUT</td>
<td>111</td>
</tr>
<tr>
<td>REMOTE I/O CONNECTOR PINOUT</td>
<td>112</td>
</tr>
<tr>
<td>AUX 1 AND AUX 2 PORTS CONNECTOR PINOUT</td>
<td>113</td>
</tr>
<tr>
<td>SCSI INTERFACE</td>
<td>115</td>
</tr>
<tr>
<td>DSCS INTERFACE</td>
<td>116</td>
</tr>
<tr>
<td>G.703 CO–DIRECTIONAL INTERFACE MODULE</td>
<td>117</td>
</tr>
<tr>
<td>I–INTERFACE I/O MODULE</td>
<td>119</td>
</tr>
</tbody>
</table>
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 Interlace parameter. Press the function key that corresponds to the type of interface you are going to use.
   a. If you selected Primary (Primary Rate Interlace) or Basic (Basic Rate Interlace) as your Physical Interlace, press F7 Physic 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 Buff (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 details 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 Setup 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 setup 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
APPLICATIONS SELECTION MENU

Applications Selection Menu

Select Application and Port

Acquisition Mode

Running

Monitoring | Ports | Monitoring | Ports
---|---|---|---
DIRTDSK | | ANALYSIS | A
TRIGGER | | X25STAT | A
DUALLINE | | BASIC | |

Simulation | Ports | Simulation | Ports
---|---|---|---
X25TEST | | BASIC | |
FR HDLC | | | |
SITREX | | | |

Select Port | FUNCTIONS
---|---
Load A | Reset | Menu | Save | Set T.O | Exit

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.

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

Menu:
Displays the main configuration menu without stopping applications.

Save:
Saves the displayed configuration to a named file.

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 configuration menu.

Set T.O:
Sets the value of the autoboot timer.
# PAGE MANIPULATION KEYS

<table>
<thead>
<tr>
<th>KEY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move ↑</td>
<td>Moves the page banner upward one line at a time (increases the size of the page).</td>
</tr>
<tr>
<td>Move ↓</td>
<td>Moves the page banner downward one line at a time (decreases the size of the page).</td>
</tr>
<tr>
<td>Scroll ↑</td>
<td>Scrolls the data displayed in the page upward one line at a time.</td>
</tr>
<tr>
<td>Scroll ↓</td>
<td>Scrolls the data displayed in the page downward one line at a time.</td>
</tr>
<tr>
<td>Shift Scroll ↑</td>
<td>Scrolls the data displayed in the page upward the number of lines displayed in the page.</td>
</tr>
<tr>
<td>Shift Scroll ↓</td>
<td>Scrolls the data displayed in the page downward the number of lines displayed in the page.</td>
</tr>
<tr>
<td>Shift Hide Page</td>
<td>Hides the active page so that the banner is no longer visible on the screen (the application continues to run).</td>
</tr>
<tr>
<td>Show Page</td>
<td>Displays a page that has been hidden with Shift Hide Page.</td>
</tr>
<tr>
<td>Replace</td>
<td>Replaces the active page with one that has been hidden using Shift Hide Page.</td>
</tr>
<tr>
<td>Shift Move ↑</td>
<td>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.</td>
</tr>
<tr>
<td>Shift Move ↓</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 determines 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

ROOT DIRECTORY (I)

TEKELEC

SIMUL
Simulation System Files

ANALYSIS

DATA
Help Files

MAN

SETUP

SYSTEM
System Files

SIMULATE
User Program Files

UTIL
System Files

D2D
Direct-to-Disk Files

HIST
History Files

MENU
Menu Config Files

UTILITY
Utility Config Files

TRIGGER
Triggering Files

BISYNC

ASYNC

APPL

(Non-protocol specific Analysis files)

FDMI

FBOP

FRAMEM

DMI Files

FRAMEM

HDMC/SDLC

PNSS

SHDLC

SIMPL

HDLQ Files

ISDN

SIMPL

LAPD Files

PSH

SLAPD

V.120

SITREX

 files

APPL

FKEY

Function Key Files

DDCMP

CUSR

Protocol-specific Analysis Files

(Files are used based on the protocol selection in the main configuration 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 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 subdirectories 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 Files**

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.

_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 any time by pressing \textit{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

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

<table>
<thead>
<tr>
<th>IF THE CHAMELEON IS TRANSMITTING FILES:</th>
<th>IF THE CHAMELEON IS RECEIVING FILES:</th>
</tr>
</thead>
<tbody>
<tr>
<td>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</td>
<td>a. Enter the host command that transmits the files. For example: send filename.ext &lt;RETURN&gt;</td>
</tr>
<tr>
<td></td>
<td>[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]</td>
</tr>
<tr>
<td>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.)</td>
<td>b. Make the File Management page active.</td>
</tr>
<tr>
<td>c. Enter the following command on the host computer: receive &lt;RETURN&gt;</td>
<td>c. Press F8 RX File. This begins reception.</td>
</tr>
<tr>
<td>d. In the Chameleon 32 File Management menu, press F7 TX File. This begins the transmission.</td>
<td></td>
</tr>
</tbody>
</table>

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

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 A-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.

After making selections Press GO
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

#### 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.

---

<table>
<thead>
<tr>
<th>Mode</th>
<th>Simulate</th>
<th>Framing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Coding</td>
<td>B8ZS</td>
<td>Channel [1..24]</td>
</tr>
<tr>
<td>Signaling</td>
<td>On</td>
<td>Channel [1..24]</td>
</tr>
<tr>
<td>Transmit Mode</td>
<td>Idle</td>
<td>00</td>
</tr>
<tr>
<td>Data Rate</td>
<td>64Kbps</td>
<td>00</td>
</tr>
<tr>
<td>Idle Channel (LSB...MSB)</td>
<td>01010101</td>
<td>00</td>
</tr>
<tr>
<td>Idle Signal (AB)</td>
<td>01</td>
<td>00</td>
</tr>
</tbody>
</table>

After making selections Press GO

---

<table>
<thead>
<tr>
<th>D4</th>
<th>ESF</th>
<th>SL96</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
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 DSO 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

Setup Menu

Device:
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: nnnnnnnnn

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 cannot 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

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
- 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

**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
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.

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

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:ITEKELEC\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:ITEKELEC\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.

<table>
<thead>
<tr>
<th>KEY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>←</td>
<td>The left arrow displays the oldest events in the buffer.</td>
</tr>
<tr>
<td>→</td>
<td>The right arrow displays the most recent events in the buffer.</td>
</tr>
<tr>
<td>↑</td>
<td>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.</td>
</tr>
<tr>
<td>↓</td>
<td>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.</td>
</tr>
<tr>
<td>Space bar</td>
<td>Stops scrolling.</td>
</tr>
<tr>
<td>Scroll ↑</td>
<td>The Scroll ↑ key moves data up one line each time you press the key.</td>
</tr>
<tr>
<td>Shift Scroll ↑</td>
<td>Shift Scroll ↑ displays the next page of data.</td>
</tr>
<tr>
<td>Scroll ↓</td>
<td>The Scroll ↓ key moves data down one line each time you press the key.</td>
</tr>
<tr>
<td>Shift Scroll ↓</td>
<td>Shift Scroll ↓ displays the previous page of data.</td>
</tr>
<tr>
<td>0-9</td>
<td>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.</td>
</tr>
<tr>
<td>F or f</td>
<td>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.</td>
</tr>
<tr>
<td>U or u</td>
<td>Un-freeze – terminates Freeze Mode and returns you to the normal History display. When unfrozen the History page displays data from the acquisition buffer.</td>
</tr>
<tr>
<td>:jump n</td>
<td>Jumps to event number n. For example, :jump 150 displays event 150 as the first event on the page. :jump 9999 displays the end of the buffer (most recent events).</td>
</tr>
<tr>
<td>:normal</td>
<td>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.</td>
</tr>
<tr>
<td>:trigger</td>
<td>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.</td>
</tr>
</tbody>
</table>
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-defined 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 rate of 98.4 in every 100,000 bits.
  - **F6 1.00E-4**: Errors inserted at the rate of 10 in every 100,000 bits.
  - **F7 1.00E-5**: Errors inserted at 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 – 34k 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 block** 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 than 10E-3.

**Consecutively Severely Error Seconds:**
This field displays the number of consecutive seconds with bit error rates worse than 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.
**DIRECT-TO-DISK APPLICATION (CONTINUED)**

**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*. 

---
The Statistics application is available for the protocols listed below.

<table>
<thead>
<tr>
<th>PROTOCOL</th>
<th>APPLICATION NAME (IN MENU)</th>
<th>STATISTICS PAGES AVAILABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSC</td>
<td>BSCSTAT</td>
<td>BSC Line Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BSC CU Statistics</td>
</tr>
<tr>
<td>ISDN</td>
<td>Q921STAT</td>
<td>Q.921 Line Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q.921 SAPI 0 Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q.921 SAPI 16 Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q.921 SAPI 83 Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q.921 Other SAPI Statistics</td>
</tr>
<tr>
<td>Primary Rate Interface</td>
<td>PRISTAT</td>
<td>PRI Error Statistics</td>
</tr>
<tr>
<td>SNA</td>
<td>SNASTAT</td>
<td>SNA Session Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDLC Line Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Session PU Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SNA LU Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDLC PU Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SNA LU Line</td>
</tr>
<tr>
<td>SS#7</td>
<td>SS7STAT</td>
<td>SS7 Line Statistics</td>
</tr>
<tr>
<td>X.25</td>
<td>X25STAT</td>
<td>X.25 Line Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HDLC Line Statistics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X.25 LCN Statistics</td>
</tr>
</tbody>
</table>

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:

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.
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.

The current call status is highlighted.

The most recent packet received from highlighted address.

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

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

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.

Prints a statistical report of 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>1-4 Conditions</th>
<th>1-4 Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4 character</td>
<td>1st Time</td>
<td>Selected by function</td>
<td>Executed if all trigger</td>
</tr>
<tr>
<td>default or</td>
<td>Enable to fire</td>
<td>keys. (See below</td>
<td>conditions are met.</td>
</tr>
<tr>
<td>user-specified name</td>
<td>Disable</td>
<td>and next page.)</td>
<td>Selected by function</td>
</tr>
<tr>
<td></td>
<td>Whenever</td>
<td></td>
<td>keys. (See below.)</td>
</tr>
</tbody>
</table>

TRIGGERING LOGIC

Condition 1 AND Condition 2 AND Condition 3 AND Condition 4

OR

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.

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-defined data string in a frame.
FrameLen Triggers on frame length in bytes.
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.
VArith 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)

<table>
<thead>
<tr>
<th>DTE/DCE</th>
<th>Error</th>
<th>Counter</th>
<th>Timer</th>
<th>RLTime</th>
<th>FramLen</th>
<th>Frame</th>
<th>Protocol</th>
<th>Variable</th>
<th>Leads</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTE</td>
<td>CRC</td>
<td>Define</td>
<td>Define</td>
<td>&gt; = or</td>
<td>&gt; =</td>
<td>&gt; =</td>
<td>$7</td>
<td>Select</td>
<td>Port A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>user</td>
<td>user</td>
<td>&lt; =</td>
<td>&lt; =</td>
<td>&lt; =</td>
<td>$8</td>
<td>&lt;</td>
<td>Port B</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>counter</td>
<td>timer</td>
<td>specified</td>
<td>specified</td>
<td>specified</td>
<td>$9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCE</td>
<td>FAabort</td>
<td>Specify</td>
<td>1,2,3,</td>
<td>1,2,3,</td>
<td>1,2,3,</td>
<td>1,2,3,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>system</td>
<td>or 4</td>
<td>or 4</td>
<td>or 4</td>
<td>or 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Float  Const  $7  $8  $9

Starting type

AscStr  EbcStr  Hex  Dec  Binary  Variable

F1  F2  F3  F4  F5  F6  F7  F8  F9

HDLC  X.25  Q.921  Q.931  DASS 2  #7 L2  #7 L3L4  TUP*  ISUP*

- Frame
- I-Frame
- RR
- RNR
- REJ
- Call Req
- Call Con
- Ctr Req
- Ctr Con
- Data
- Intrupt
- Intrupt C
- RRP
- RNR
- REJ
- P/F
- SAP1
- TEI
- C/R

- ProDisc
- CR Flag
- CR Value
- Msg Type
- Message
- Type
- LAP
- (time slot)
- FSN
- BSN
- FIB
- BIB
- FIN
- LSSU
- MSU
- LSSU SF

* 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.

<table>
<thead>
<tr>
<th>Standard</th>
<th>F8</th>
<th>F9</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCITT (84 or 88)</td>
<td>CT TUP</td>
<td>CT ISUP</td>
</tr>
<tr>
<td>NCC</td>
<td>NCC TUP</td>
<td>NCCISUP</td>
</tr>
<tr>
<td>NTT</td>
<td>NTT TUP</td>
<td>NTTISUP</td>
</tr>
<tr>
<td>ANSI</td>
<td>ASN TUP</td>
<td>ANSIISUP</td>
</tr>
<tr>
<td>ITR7</td>
<td>TR7 TUP</td>
<td>TR7ISUP</td>
</tr>
</tbody>
</table>

- High
- Low
- To High
- To Low

-32-
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

<table>
<thead>
<tr>
<th>If you are here:</th>
<th>And you want to:</th>
<th>Do this:</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRAMEM, SIMP/L BSC, or Async simulation prompt!</td>
<td>Stop the simulator</td>
<td>Enter: MENU &lt;RETURN&gt; 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.</td>
</tr>
<tr>
<td></td>
<td>Access the parameter set-up menu</td>
<td>Enter: SETUP &lt;RETURN&gt;</td>
</tr>
<tr>
<td></td>
<td>Write programs</td>
<td>Read the Chameleon 32 Simulation Manual (Chapter 3) for Chameleon BASIC fundamentals AND read the chapter that describes your simulation language.</td>
</tr>
<tr>
<td>SITREX SIMULATOR ACTIVE</td>
<td>Return to the main menu</td>
<td>Enter: PS &lt;RETURN&gt; HALT &lt;RETURN&gt;</td>
</tr>
<tr>
<td></td>
<td>Access the parameter set-up menu</td>
<td>Enter: PS &lt;RETURN&gt; HALT &lt;RETURN&gt; Press: F2 GO</td>
</tr>
<tr>
<td></td>
<td>Enter command mode (!)</td>
<td>Enter: PS &lt;RETURN&gt;</td>
</tr>
<tr>
<td></td>
<td>Exit command mode</td>
<td>Enter: EXIT &lt;RETURN&gt;</td>
</tr>
<tr>
<td></td>
<td>Activate the trace buffer</td>
<td>Enter: PP &lt;RETURN&gt;</td>
</tr>
<tr>
<td></td>
<td>Deactivate the trace buffer</td>
<td>Enter: CTRL P</td>
</tr>
<tr>
<td>Any Parameter Set-Up Menu</td>
<td>Access the simulation prompt (!) to write programs</td>
<td>Press: Z</td>
</tr>
<tr>
<td></td>
<td>Return to the main menu</td>
<td>Press: ESC</td>
</tr>
<tr>
<td></td>
<td>Change parameter values</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Save parameter values</td>
<td>Press: S</td>
</tr>
</tbody>
</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.

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>I-FIELD</th>
<th>DECIMAL</th>
<th>HEX</th>
<th>BINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFRAME</td>
<td>0</td>
<td>00</td>
<td>00</td>
<td>00000000</td>
</tr>
<tr>
<td>SNRME</td>
<td>207</td>
<td>CF</td>
<td>11001111</td>
<td></td>
</tr>
<tr>
<td>SARME</td>
<td>79</td>
<td>4F</td>
<td>01001111</td>
<td></td>
</tr>
<tr>
<td>SABME</td>
<td>111</td>
<td>6F</td>
<td>01101111</td>
<td></td>
</tr>
<tr>
<td>SREJ</td>
<td>13</td>
<td>0D</td>
<td>00001101</td>
<td></td>
</tr>
<tr>
<td>SNRM</td>
<td>131</td>
<td>83</td>
<td>10000011</td>
<td></td>
</tr>
<tr>
<td>SARM</td>
<td>15</td>
<td>0F</td>
<td>00001111</td>
<td></td>
</tr>
<tr>
<td>SABM</td>
<td>47</td>
<td>2F</td>
<td>00101111</td>
<td></td>
</tr>
<tr>
<td>DISC</td>
<td>67</td>
<td>43</td>
<td>01000011</td>
<td></td>
</tr>
<tr>
<td>RSET</td>
<td>143</td>
<td>8F</td>
<td>10000111</td>
<td></td>
</tr>
<tr>
<td>FRMR</td>
<td>135</td>
<td>87</td>
<td>10000111</td>
<td></td>
</tr>
<tr>
<td>TEST</td>
<td>227</td>
<td>E3</td>
<td>11100001</td>
<td></td>
</tr>
<tr>
<td>CMDR</td>
<td>135</td>
<td>87</td>
<td>10000111</td>
<td></td>
</tr>
<tr>
<td>RNR</td>
<td>5</td>
<td>05</td>
<td>00000101</td>
<td></td>
</tr>
<tr>
<td>REJ</td>
<td>9</td>
<td>09</td>
<td>00001001</td>
<td></td>
</tr>
<tr>
<td>SIM</td>
<td>7</td>
<td>07</td>
<td>00000111</td>
<td></td>
</tr>
<tr>
<td>XID</td>
<td>175</td>
<td>AF</td>
<td>10101111</td>
<td></td>
</tr>
<tr>
<td>RIM</td>
<td>7</td>
<td>07</td>
<td>00000111</td>
<td></td>
</tr>
<tr>
<td>NSI</td>
<td>3</td>
<td>03</td>
<td>00000111</td>
<td></td>
</tr>
<tr>
<td>RQI</td>
<td>7</td>
<td>07</td>
<td>00000111</td>
<td></td>
</tr>
<tr>
<td>ROL</td>
<td>15</td>
<td>0F</td>
<td>00001111</td>
<td></td>
</tr>
<tr>
<td>NSP</td>
<td>35</td>
<td>23</td>
<td>00100011</td>
<td></td>
</tr>
<tr>
<td>RR</td>
<td>1</td>
<td>01</td>
<td>00000001</td>
<td></td>
</tr>
<tr>
<td>UI</td>
<td>3</td>
<td>03</td>
<td>00000011</td>
<td></td>
</tr>
<tr>
<td>UP</td>
<td>35</td>
<td>23</td>
<td>00100011</td>
<td></td>
</tr>
<tr>
<td>DM</td>
<td>15</td>
<td>0F</td>
<td>00001111</td>
<td></td>
</tr>
<tr>
<td>UA</td>
<td>99</td>
<td>63</td>
<td>01100011</td>
<td></td>
</tr>
<tr>
<td>RD</td>
<td>67</td>
<td>43</td>
<td>01000011</td>
<td></td>
</tr>
</tbody>
</table>
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.

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

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

### SIMP/L HDLC DEFAULT MNEMONIC TABLE

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>FIELD WIDTH (BITS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCGN</td>
<td>4</td>
</tr>
<tr>
<td>PKID</td>
<td>8</td>
</tr>
<tr>
<td>P(S)</td>
<td>3</td>
</tr>
<tr>
<td>P(R)</td>
<td>3</td>
</tr>
<tr>
<td>PAD1</td>
<td>1</td>
</tr>
<tr>
<td>MBIT</td>
<td>1</td>
</tr>
<tr>
<td>DBIT</td>
<td>1</td>
</tr>
<tr>
<td>QBIT</td>
<td>1</td>
</tr>
<tr>
<td>PAD2</td>
<td>2</td>
</tr>
<tr>
<td>GFI</td>
<td>4</td>
</tr>
<tr>
<td>LCN</td>
<td>8</td>
</tr>
</tbody>
</table>
# BSC DEFAULT MNEMONIC TABLE

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>DECIMAL</th>
<th>HEX</th>
<th>BINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK0</td>
<td>112</td>
<td>70</td>
<td>01110000</td>
</tr>
<tr>
<td>ACK1</td>
<td>97</td>
<td>61</td>
<td>01100001</td>
</tr>
<tr>
<td>WABT</td>
<td>127</td>
<td>7F</td>
<td>01111111</td>
</tr>
<tr>
<td>SOH</td>
<td>1</td>
<td>01</td>
<td>00000001</td>
</tr>
<tr>
<td>STX</td>
<td>2</td>
<td>02</td>
<td>00000010</td>
</tr>
<tr>
<td>ETB</td>
<td>38</td>
<td>26</td>
<td>00100110</td>
</tr>
<tr>
<td>ETX</td>
<td>3</td>
<td>03</td>
<td>00000011</td>
</tr>
<tr>
<td>ITB</td>
<td>31</td>
<td>1F</td>
<td>00011111</td>
</tr>
<tr>
<td>EOT</td>
<td>55</td>
<td>37</td>
<td>00110111</td>
</tr>
<tr>
<td>ENQ</td>
<td>45</td>
<td>2D</td>
<td>00101101</td>
</tr>
<tr>
<td>DLE</td>
<td>16</td>
<td>10</td>
<td>00010000</td>
</tr>
<tr>
<td>SYN</td>
<td>50</td>
<td>32</td>
<td>00110010</td>
</tr>
<tr>
<td>ACK</td>
<td>46</td>
<td>2E</td>
<td>00101110</td>
</tr>
<tr>
<td>NAK</td>
<td>61</td>
<td>3D</td>
<td>00111101</td>
</tr>
<tr>
<td>MNEMONIC</td>
<td>DECIMAL</td>
<td>HEX</td>
<td>BINARY</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>------</td>
<td>--------------</td>
</tr>
<tr>
<td>SPACE</td>
<td>32</td>
<td>20</td>
<td>00100000</td>
</tr>
<tr>
<td>BELL</td>
<td>7</td>
<td>07</td>
<td>00000111</td>
</tr>
<tr>
<td>NULL</td>
<td>0</td>
<td>00</td>
<td>00000000</td>
</tr>
<tr>
<td>SOH</td>
<td>1</td>
<td>01</td>
<td>00000001</td>
</tr>
<tr>
<td>STX</td>
<td>2</td>
<td>02</td>
<td>00000010</td>
</tr>
<tr>
<td>ETX</td>
<td>3</td>
<td>03</td>
<td>00000011</td>
</tr>
<tr>
<td>EOT</td>
<td>4</td>
<td>04</td>
<td>00000100</td>
</tr>
<tr>
<td>ENQ</td>
<td>5</td>
<td>05</td>
<td>00000101</td>
</tr>
<tr>
<td>ACK</td>
<td>6</td>
<td>06</td>
<td>00000110</td>
</tr>
<tr>
<td>DLE</td>
<td>16</td>
<td>10</td>
<td>00010000</td>
</tr>
<tr>
<td>DC1</td>
<td>17</td>
<td>11</td>
<td>00010001</td>
</tr>
<tr>
<td>DC2</td>
<td>18</td>
<td>12</td>
<td>00010010</td>
</tr>
<tr>
<td>DC3</td>
<td>19</td>
<td>13</td>
<td>00010011</td>
</tr>
<tr>
<td>DC4</td>
<td>20</td>
<td>14</td>
<td>00010100</td>
</tr>
<tr>
<td>NAK</td>
<td>21</td>
<td>15</td>
<td>00010101</td>
</tr>
<tr>
<td>SYN</td>
<td>22</td>
<td>16</td>
<td>00010110</td>
</tr>
<tr>
<td>ETB</td>
<td>23</td>
<td>17</td>
<td>00010111</td>
</tr>
<tr>
<td>CAN</td>
<td>24</td>
<td>18</td>
<td>00011000</td>
</tr>
<tr>
<td>SUB</td>
<td>26</td>
<td>1A</td>
<td>00011010</td>
</tr>
<tr>
<td>ESC</td>
<td>27</td>
<td>1B</td>
<td>00011011</td>
</tr>
<tr>
<td>DEL</td>
<td>127</td>
<td>7F</td>
<td>01111111</td>
</tr>
<tr>
<td>BS</td>
<td>8</td>
<td>08</td>
<td>00001000</td>
</tr>
<tr>
<td>HT</td>
<td>9</td>
<td>09</td>
<td>00001001</td>
</tr>
<tr>
<td>LF</td>
<td>10</td>
<td>0A</td>
<td>00001010</td>
</tr>
<tr>
<td>VT</td>
<td>11</td>
<td>0B</td>
<td>00001011</td>
</tr>
<tr>
<td>FF</td>
<td>12</td>
<td>0C</td>
<td>00001100</td>
</tr>
<tr>
<td>CR</td>
<td>13</td>
<td>0D</td>
<td>00001101</td>
</tr>
<tr>
<td>SO</td>
<td>14</td>
<td>0E</td>
<td>00001110</td>
</tr>
<tr>
<td>SI</td>
<td>15</td>
<td>0F</td>
<td>00001111</td>
</tr>
<tr>
<td>EM</td>
<td>25</td>
<td>19</td>
<td>00011001</td>
</tr>
<tr>
<td>FS</td>
<td>28</td>
<td>1C</td>
<td>00011100</td>
</tr>
<tr>
<td>GS</td>
<td>29</td>
<td>1D</td>
<td>00011101</td>
</tr>
<tr>
<td>RS</td>
<td>30</td>
<td>1E</td>
<td>00011110</td>
</tr>
<tr>
<td>US</td>
<td>31</td>
<td>1F</td>
<td>00011111</td>
</tr>
</tbody>
</table>
BASIC COMMANDS

@ References the array.  @exp -array subscript

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

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

ATIMEx Returns the ASCII value of the realtime
ATIMEx stamp.

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

BCDS$ ASCII to BCD conversion.
BCDS$(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"

CHRS Assigns the binary equivalent of an ASCII
CHRS(x)

CLEAR Clears the trace buffer.
CLEAR

CLOSE Closes all open files.
CLOSE 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.
$X = DEC$(x)

DEFINE Defines a mnemonic for the mnemonic
table. Syntax is protocol-specific.
DEFINE "name"

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

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

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

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
EOF of data file is reached.
EOF = 0
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^A
^x =function key (1 - 10)
^A =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
FLOAD  Loads a function key definition file into memory.
FLUSH  Clears the acquisition buffer.
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
HEXS    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)
HLFUND  Displays text in double intensity and underlined.
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
INKEYS Assigns the next character typed on the keyboard to a string variable.
$A=INKEYS$
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 $A
INS     Inserts a blank line on the screen.
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
LEFTS   Assigns a specified number of characters from the left end of one string to
another string.
$A = LEFTS($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
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST</td>
<td>Displays program in memory. Lists entire program.</td>
</tr>
<tr>
<td>LIST x</td>
<td>Lists program from line x to end.</td>
</tr>
<tr>
<td>LIST x,y</td>
<td>Lists program from line x to line y.</td>
</tr>
<tr>
<td>LIST,y</td>
<td>Lists program from beginning to line y.</td>
</tr>
<tr>
<td>LLIST</td>
<td>Outputs program in memory to a printer.</td>
</tr>
<tr>
<td>LLIST x</td>
<td>Prints program from line x to end.</td>
</tr>
<tr>
<td>LLIST x,y</td>
<td>Prints program from line x to line y.</td>
</tr>
<tr>
<td>LLIST ,y</td>
<td>Prints program from beginning to line y.</td>
</tr>
<tr>
<td>LMLIST</td>
<td>Outputs the mnemonic table in memory to a printer.</td>
</tr>
<tr>
<td>LOAD</td>
<td>Loads a program file into memory.</td>
</tr>
<tr>
<td>LPRINT</td>
<td>Outputs the contents of the trace buffer to a printer.</td>
</tr>
<tr>
<td>MENU</td>
<td>Exits the simulator and returns to the main menu.</td>
</tr>
<tr>
<td>MERGE</td>
<td>Combines a program file with the program in memory.</td>
</tr>
<tr>
<td>MLOAD</td>
<td>Loads a mnemonic table into memory.</td>
</tr>
<tr>
<td>MSAVE</td>
<td>Saves the mnemonic table in memory to disk.</td>
</tr>
<tr>
<td>NEW</td>
<td>Deletes the program in memory.</td>
</tr>
<tr>
<td>NEXT</td>
<td>Increments the counter in a FOR loop.</td>
</tr>
<tr>
<td>NRM</td>
<td>Cancels display effects commands (blinking, underline, double intensity).</td>
</tr>
<tr>
<td>OPEN</td>
<td>Opens a data file.</td>
</tr>
<tr>
<td>OPEN &quot;I&quot;,</td>
<td>Opens a file for input.</td>
</tr>
<tr>
<td>&quot;filename&quot;</td>
<td></td>
</tr>
<tr>
<td>OPEN &quot;O&quot;,</td>
<td>Opens a new file for output.</td>
</tr>
<tr>
<td>&quot;filename&quot;</td>
<td></td>
</tr>
<tr>
<td>OPEN &quot;A&quot;,</td>
<td>Opens file for output to the end of the file.</td>
</tr>
<tr>
<td>&quot;filename&quot;</td>
<td></td>
</tr>
<tr>
<td>PRINT</td>
<td>Displays a string, expression, or variable.</td>
</tr>
<tr>
<td>PRINT &quot;string&quot;</td>
<td>Prints the string.</td>
</tr>
<tr>
<td>PRINT $A</td>
<td>Prints string variable.</td>
</tr>
<tr>
<td>PRINT x</td>
<td>Prints numeric variable.</td>
</tr>
<tr>
<td>PRINT $%x</td>
<td>Prints x in hex.</td>
</tr>
<tr>
<td>READ</td>
<td>Reads next record from an input file.</td>
</tr>
<tr>
<td>READ $A</td>
<td></td>
</tr>
<tr>
<td>REC</td>
<td>Protocol-specific command that transfers data from the acquisition buffer to the trace buffer.</td>
</tr>
<tr>
<td>REM</td>
<td>Programmer's internal remark.</td>
</tr>
<tr>
<td>REM comment</td>
<td></td>
</tr>
<tr>
<td>RESEQ</td>
<td>Re-numbers the line numbers of the program in memory.</td>
</tr>
<tr>
<td>RESEQ Start at line 10, increment by 10.</td>
<td></td>
</tr>
<tr>
<td>RESEQ (EXPR1)</td>
<td>Start at x, increment by 10.</td>
</tr>
<tr>
<td>RESEQ (EXPR1) {EXPR2}</td>
<td>Start at x, increment by y.</td>
</tr>
<tr>
<td>RETURN</td>
<td>Returns program control from a subroutine called by a GOSUB.</td>
</tr>
<tr>
<td>REV</td>
<td>Displays text in reverse video.</td>
</tr>
<tr>
<td>REVHLF</td>
<td>Displays text in reverse video in double intensity.</td>
</tr>
<tr>
<td>REVUND</td>
<td>Displays text in reverse video and underlined.</td>
</tr>
<tr>
<td>RIGHTS$</td>
<td>Assigns a specified number of characters from the right end of one string to another string.</td>
</tr>
<tr>
<td>RND</td>
<td>Returns a random number.</td>
</tr>
<tr>
<td>RND(x)</td>
<td></td>
</tr>
</tbody>
</table>
### BASIC COMMANDS

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RUN</strong></td>
<td>Executes the program in memory.</td>
</tr>
<tr>
<td><strong>SAVE</strong></td>
<td>Saves the program in memory to disk.</td>
</tr>
<tr>
<td><strong>SET</strong></td>
<td>Sets physical interface signal to 1 or 0. Not available in SIMPL or FRAMEM DMI.</td>
</tr>
<tr>
<td><strong>SETUP</strong></td>
<td>Accesses the parameter set-up menu. Not available in FRAMEM DMI.</td>
</tr>
<tr>
<td><strong>SIZE</strong></td>
<td>Returns size of free program area in bytes.</td>
</tr>
<tr>
<td><strong>STOP</strong></td>
<td>Terminates program execution.</td>
</tr>
<tr>
<td><strong>TEST</strong></td>
<td>Tests an interface signal for 1 or 0.</td>
</tr>
<tr>
<td><strong>TFREE</strong></td>
<td>Returns the length of the unused trace buffer in bytes.</td>
</tr>
<tr>
<td><strong>TIM0</strong></td>
<td>Timer which counts down in ten millisecond (.01) intervals</td>
</tr>
<tr>
<td><strong>TIM1</strong></td>
<td>Timer which counts up in ten millisecond (.01) intervals</td>
</tr>
<tr>
<td><strong>TIM2</strong></td>
<td>Timer which counts down in seconds</td>
</tr>
<tr>
<td><strong>TIM3</strong></td>
<td>Timer which counts up in seconds</td>
</tr>
<tr>
<td><strong>TIME</strong></td>
<td>Read-only variable that returns a specified byte of the system time in BCD digits.</td>
</tr>
<tr>
<td><strong>TLOAD</strong></td>
<td>Loads a trace file into memory.</td>
</tr>
<tr>
<td><strong>TPRINT</strong></td>
<td>Displays the contents of the trace buffer.</td>
</tr>
<tr>
<td><strong>TROFF</strong></td>
<td>Turns off the program trace facility (debug mode).</td>
</tr>
<tr>
<td><strong>TRON</strong></td>
<td>Turns on the program trace facility (debug mode).</td>
</tr>
<tr>
<td><strong>TSAVE</strong></td>
<td>Saves the contents of the trace buffer to a trace file.</td>
</tr>
<tr>
<td><strong>UND</strong></td>
<td>Displays text in underline.</td>
</tr>
<tr>
<td><strong>VAL</strong></td>
<td>Converts a numeric ASCII string to its integer form.</td>
</tr>
<tr>
<td><strong>WRITE</strong></td>
<td>Writes a string variable to a data file opened for output.</td>
</tr>
<tr>
<td><strong>XYPLOT</strong></td>
<td>Moves the cursor to a specified position on the screen.</td>
</tr>
</tbody>
</table>

- **TIM2** = \( x \) (Timer which counts down in seconds)
- **TIM3** = \( x \) (Timer which counts up in seconds)
- **TIME** = \( x \) (Read-only variable that returns a specified byte of the system time in BCD digits)
- **TLOAD** = "filename"
- **TPRINT** = "filename"
- **TROFF**
- **TRON**
- **TSAVE** = "filename"
- **UND**
- **WRITE** = \( \text{VAL}(\$A) \)
- **WRITE** = \( \$A \)
<table>
<thead>
<tr>
<th>FRAMEM COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ABORTRAN</strong></td>
</tr>
<tr>
<td>Transmits a frame with an abort sequence. The frame must be greater than 4 bytes in length.</td>
</tr>
<tr>
<td><strong>BADTRAN</strong></td>
</tr>
<tr>
<td>Transmits a frame with a bad CRC. The frame must be greater than 4 bytes in length.</td>
</tr>
<tr>
<td><strong>CRC</strong></td>
</tr>
<tr>
<td>Indicates if received frame had a good or bad CRC.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>DEFINE</strong></td>
</tr>
<tr>
<td>Defines new mnemonics or redefines existing mnemonics.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>DEFSUB</strong></td>
</tr>
<tr>
<td>Defines the line number to jump to when the received frame matches a specific mnemonic.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>EXTEND</strong></td>
</tr>
<tr>
<td>Selects extended mode addressing.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>GET</strong></td>
</tr>
<tr>
<td>Gets two bytes (low byte, high byte) from an I-field.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>MOD</strong></td>
</tr>
<tr>
<td>Specifies modulo 8 or modulo 128 sequencing.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>NORM</strong></td>
</tr>
<tr>
<td>Selects normal mode addressing.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>PUT</strong></td>
</tr>
<tr>
<td>Defines a specified byte in an I-field for transmission.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>REC</strong></td>
</tr>
<tr>
<td>Assigns the next received frame in sequence from the acquisition buffer and to 0 or more string variables.</td>
</tr>
</tbody>
</table>

| **RXADDR**       |
| Address field of the received frame. |
| 
| **RXC/R**        |
| C/R bit extracted from an FRMR field. |
| 
| **RXDIAG**       |
| Last byte of an FRMR (WXYZ bits). |
| 
| **RXFCTL**       |
| Control field of the received frame without the poll/final bit, N(S), and N(R). |
| 
| **RXFRLEN**      |
| Length of the received frame. |
| 
| **RXRP/F**       |
| Poll/final bit of a received rejected frame control field. |
| 
| **RXV(R)**       |
| V(R) of the rejecting station for a rejected frame. |
| 
| **RXV(S)**       |
| V(S) of the rejecting station for a rejected frame. |
| 
| **STATUS**       |
| Displays the current addressing mode and modulo. |
| 
| **TPRINT**       |
| Displays the contents of the trace buffer. |
FRAMEM COMMANDS

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAN</td>
<td>Transmits a frame with a good CRC.</td>
</tr>
<tr>
<td>TXADD</td>
<td>Sets the value of the address field of the frame being transmitted.</td>
</tr>
<tr>
<td>TXC/R</td>
<td>Sets the value of the C/R bit of the FRMR frame being transmitted.</td>
</tr>
<tr>
<td>TXDIAG</td>
<td>Sets the value of the last byte (WXYZ bits) of an FRMR field.</td>
</tr>
<tr>
<td>TXFCTL</td>
<td>Sets the value of the frame control field of the frame being transmitted.</td>
</tr>
<tr>
<td>TXIFIELD</td>
<td>Sets or adds to the contents of an I-field for a frame being transmitted.</td>
</tr>
<tr>
<td>TXN(R)</td>
<td>Sets the value of N(R) of the frame being transmitted.</td>
</tr>
<tr>
<td>TXN(S)</td>
<td>Sets the value of N(S) of the frame being transmitted.</td>
</tr>
<tr>
<td>TXP/F</td>
<td>Sets the poll/final bit of the frame being transmitted.</td>
</tr>
<tr>
<td>TXRFCTL</td>
<td>Sets the rejected frame control field of a frame being transmitted.</td>
</tr>
<tr>
<td>TXV(R)</td>
<td>Sets the value of V(R) for the frame being transmitted.</td>
</tr>
<tr>
<td>TXV(S)</td>
<td>Sets the value of V(S) for the frame being transmitted.</td>
</tr>
</tbody>
</table>

FRAMEM LAPD COMMANDS AND VARIABLES

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILL</td>
<td>Changes the interframe fill pattern.</td>
</tr>
<tr>
<td>RXCR</td>
<td>C/R bit of the received frame.</td>
</tr>
<tr>
<td>RXSAPI</td>
<td>SAPI of the received frame.</td>
</tr>
<tr>
<td>RXTEI</td>
<td>TEI of the received frame.</td>
</tr>
<tr>
<td>TXCR</td>
<td>Sets the value of the C/R bit of the frame being transmitted.</td>
</tr>
<tr>
<td>TXSAPI</td>
<td>Sets the value of the SAPI for the frame being transmitted.</td>
</tr>
<tr>
<td>TXTEI</td>
<td>Sets the TEI for the frame being transmitted.</td>
</tr>
</tbody>
</table>
# FRAMEM DMI Commands

<table>
<thead>
<tr>
<th>FRAMEM DMI Commands and Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CAUSE</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>CHADMIN</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>CONNECT</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>DCALLED</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>DCALLING</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>DISCONNECT</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>DMATCH</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>DTIMERS</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

## 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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREAK</td>
<td>Disassembles an I-field into its component strings and/or user-defined mnemonics.</td>
</tr>
<tr>
<td>BUFFER</td>
<td>Defines a message for the transmission buffer in hex.</td>
</tr>
<tr>
<td>BUILD</td>
<td>Assembles a message in the transmission buffer.</td>
</tr>
<tr>
<td>DEFINE</td>
<td>Defines new frame control mnemonic or redefines an existing mnemonic.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>Returns length of the received frame.</td>
</tr>
<tr>
<td>LRDISPF</td>
<td>Outputs the last data field received to a printer or remote device.</td>
</tr>
<tr>
<td>LTDISPF</td>
<td>Outputs the last data field built to a printer or remote device.</td>
</tr>
<tr>
<td>RDISPF</td>
<td>Displays the last data field received.</td>
</tr>
<tr>
<td>REC</td>
<td>Transfers the next message in sequence from the reception buffer to the trace buffer.</td>
</tr>
<tr>
<td>SLOF</td>
<td>Disconnects link by sending a DISC.</td>
</tr>
<tr>
<td>SLON</td>
<td>Attempts to set the frame level link by sending a SABM, SABME, or SNRM.</td>
</tr>
<tr>
<td>STATUS</td>
<td>Displays the status of the link.</td>
</tr>
<tr>
<td>TDISPF</td>
<td>Displays the last data field built.</td>
</tr>
<tr>
<td>TPRINT</td>
<td>Displays the trace buffer.</td>
</tr>
<tr>
<td>TRAN</td>
<td>Transmits a message.</td>
</tr>
</tbody>
</table>

### SIMP/L HDLC COMMANDS AND VARIABLES

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LNKSTAT</td>
<td>Returns the status of the link.</td>
</tr>
<tr>
<td>SET</td>
<td>Sets variable values.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSI</td>
<td>Transmits an NSI frame.</td>
</tr>
<tr>
<td>SET</td>
<td>Sets the value of variables and timers.</td>
</tr>
<tr>
<td>TEST</td>
<td>Sends a test frame.</td>
</tr>
<tr>
<td>XID</td>
<td>Transmits an XID frame.</td>
</tr>
<tr>
<td>XIDFLD</td>
<td>Sets the data field of an XID frame.</td>
</tr>
</tbody>
</table>

**Example:**
- `BREAK udm, udm, udm
  BREAK $A, $B, $C...
  BREAK udm, $A, udm, $B, udm, $C...

**Example:**
- `BUFFER = xxx
  xxx is the message`
SIMP/L COMMANDS

**SIMP/L LAPD COMMANDS AND VARIABLES**

* Extendable SIMP/L LAPD only.

*EXTEND Invokes Extendable SIMP/L LAPD.

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

LNKSTAT Returns the status of the link.

SET LNKSTAT

IN LINKSTAT...

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.

SET MODx

MODx = x Range: 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 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 {RSAPI}**

Sets the values of 1 - 3 user-defined receive SAPIs.

SET RSAPI = x

n is the SAPI number, range 0 - 2

x is the SAPI value, range 0 - 255

**SET RTEL**

Sets the values of 1 - 3 user-defined receive TEIs.

SET RTELn = x

n is the TEI number, range 0 - 2

x is the TEI value, range 0 - 255

**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 {XID}**

XID Exchange

0 = Stop transmitting XIDs on T203 timeout.

1 = Transmit XID command on T203 timeout.

**SET TRUI**

Transmits a UI frame

TRUI

**SET TRXIDC**

Transmits an XID command frame.

TRXIDC

**SET TRXIDR**

Transmits an XID response frame.

TRXIDR
## SIMP/L COMMANDS

### MULTI-LINK SIMP/L LAPD COMMANDS

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRELNK</td>
<td>Read-only variable that returns the number of the lowest disabled link.</td>
</tr>
<tr>
<td>FRSTAT</td>
<td>Read-only variable that returns a 2-byte frame status value. Interpretation of second byte:</td>
</tr>
<tr>
<td>RECLNK</td>
<td>Read-only variable that returns the number of the link from which data was last received.</td>
</tr>
<tr>
<td>SET LINK</td>
<td>Places one of the 64 available links under user control.</td>
</tr>
<tr>
<td>SET SAPI</td>
<td>Sets SAPI value for the selected link.</td>
</tr>
<tr>
<td>SET TEI</td>
<td>Sets TEI value for the selected link.</td>
</tr>
<tr>
<td>STATE</td>
<td>Displays states of all 64 links.</td>
</tr>
<tr>
<td>STATUS</td>
<td>Displays status of selected link.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LNKSTAT</td>
<td></td>
</tr>
<tr>
<td>SET N200</td>
<td></td>
</tr>
<tr>
<td>SET N201</td>
<td></td>
</tr>
<tr>
<td>SET T200</td>
<td></td>
</tr>
<tr>
<td>SET T203</td>
<td></td>
</tr>
<tr>
<td>SET CONFIG</td>
<td></td>
</tr>
<tr>
<td>TRXIDR</td>
<td></td>
</tr>
<tr>
<td>TRXIDC</td>
<td></td>
</tr>
</tbody>
</table>

### V.120 SIMP/L COMMANDS AND VARIABLES

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRELNK</td>
<td>Read-only variable that returns the number of the lowest disabled link.</td>
</tr>
<tr>
<td>FRSTAT</td>
<td>Read-only variable that returns a 2-byte frame status value. Interpretation of second byte:</td>
</tr>
<tr>
<td>RECLNK</td>
<td>Read-only variable that returns the number of the link from which data was last received.</td>
</tr>
<tr>
<td>RTRAN</td>
<td>Transmits an I-Frame response. The C/R bit is set to 0 in command frames and 1 in response frames.</td>
</tr>
<tr>
<td>SET LINK</td>
<td>Places one of the 64 available links under user control.</td>
</tr>
<tr>
<td>SET SAPI</td>
<td>Sets SAPI value for the selected link.</td>
</tr>
<tr>
<td>SET TEI</td>
<td>Sets TEI value for the selected link.</td>
</tr>
<tr>
<td>STATE</td>
<td>Displays states of all 64 links.</td>
</tr>
<tr>
<td>STATUS</td>
<td>Displays status of selected link.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LNKSTAT</td>
<td></td>
</tr>
<tr>
<td>SET N200</td>
<td></td>
</tr>
<tr>
<td>SET N201</td>
<td></td>
</tr>
<tr>
<td>SET T200</td>
<td></td>
</tr>
<tr>
<td>SET T203</td>
<td></td>
</tr>
<tr>
<td>SET CONFIG</td>
<td></td>
</tr>
<tr>
<td>TRXIDR</td>
<td></td>
</tr>
<tr>
<td>TRXIDC</td>
<td></td>
</tr>
</tbody>
</table>
**ASYNC BASIC COMMANDS**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BREAK</strong></td>
<td>Transmits a BREAK sequence.</td>
</tr>
<tr>
<td><strong>CRC16</strong></td>
<td>Calculates the CRC for a string $B=\text{CRC16}($A)</td>
</tr>
<tr>
<td><strong>FOXMESS</strong></td>
<td>Transmits the standard FOX message and repeats it until the operator hits any key.</td>
</tr>
<tr>
<td><strong>FRAMING</strong></td>
<td>Returns a value that indicates the presence of stop bits at the end of the received block.</td>
</tr>
<tr>
<td><strong>PRINT FRAMING</strong></td>
<td>IF FRAMING... FRAMING returns: FRAMING=0 Stop bits FRAMING=1 No stop bits</td>
</tr>
<tr>
<td><strong>LENGTH</strong></td>
<td>Returns the number of characters received in a block.</td>
</tr>
<tr>
<td><strong>PRINT LENGTH</strong></td>
<td>IF LENGTH...</td>
</tr>
<tr>
<td><strong>LRC</strong></td>
<td>Calculates the LRC for a string. $X = \text{LRC}(Y)$</td>
</tr>
<tr>
<td><strong>PARITY</strong></td>
<td>Indicates whether a parity error has occurred.</td>
</tr>
<tr>
<td><strong>PRINT PARITY</strong></td>
<td>IF PARITY... PARITY returns: PARITY =0 No parity error PARITY =1 Parity error</td>
</tr>
<tr>
<td><strong>REC</strong></td>
<td>Assigns the next character (if in character mode) or block of characters (if in block mode) to string variables.</td>
</tr>
<tr>
<td><strong>PRINT</strong></td>
<td></td>
</tr>
<tr>
<td><strong>TRAN</strong></td>
<td></td>
</tr>
<tr>
<td><strong>PRINT TRAN</strong></td>
<td></td>
</tr>
<tr>
<td><strong>TPRINT</strong></td>
<td>Displays the contents of the trace buffer.</td>
</tr>
<tr>
<td><strong>TPRINT</strong></td>
<td><strong>TPRINT</strong> Prints the trace in ASCII. <strong>TPRINT</strong> HEX Prints the trace in hexadecimal.</td>
</tr>
<tr>
<td><strong>TRAN</strong></td>
<td>Transmits data in strings, mnemonics or literal data.</td>
</tr>
<tr>
<td><strong>TRAN</strong></td>
<td>TRAN $A$...</td>
</tr>
<tr>
<td><strong>TRAN</strong></td>
<td>TRAN CR, LF...</td>
</tr>
<tr>
<td><strong>TRAN</strong></td>
<td>TRAN &quot;ABCD&quot;...</td>
</tr>
<tr>
<td><strong>TRAN</strong></td>
<td>TRAN $A$, CR, LF, &quot;ABCD&quot;</td>
</tr>
</tbody>
</table>
**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

```
7 6 5 4 3 2 1 0
0 = SOH
1 = STX

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

START FRAMING

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.
- `FAs...a` Frame defined in ASCII.
- `FHe...h` Frame defined in hex.

**Send Unnumbered Commands**
- `FPDISC` Polled/unpolled DISC on primary address.
- `FPSABM` Polled/unpolled SABM on primary address.

**Send Unnumbered Responses**
- `FUAA` UA frame.
- `FUOM` OM frame.
- `FUCMORlh...h(V)(,Vr)(B)(W)(X)(Y)(Z)` CMDR frame.

**Send Numbered Commands**
- `FPRR(Nr)` Sends an RR.
- `FPRNR(Nr)` Sends an RNR.

**Numbered Responses**
- `FURR(Nr)` Sends a RNR frame.
- `FURM(Nr)` Sends a RR frame.

**Send Information Frame**
- `FNS(Nr)(,Pr)(PACKET)` Packet mnemonic.
- `FNS(Nr)(,Pr)(PHh1h2...h)` Packet in hex.
- `FNS(Nr)(,Pr)(PAabcd...a)` Packet in ASCII.

**PACKET LEVEL COMMANDS**

**Send User-Defined Packet**
- `PHh...h` Packet in hex.
- `PAs...a` Packet in ASCII.

**Send Call/Call Confirmation Packets**
- `PUNCALL(D)(Na or V)(Nb),(Fhm...h),(Dhm...h)` Sends a Call packet.
- `PUNCALL(D)(Na or V)(Nb),(Fhm...h),(DAa...a)` Sends a Call Confirmation packet.

**Send Diagnostic Packets**
- `PUND(Ps)(Pr)(Q)(M)(O)Hh...h` Hex.
- `PUND(Ps)(Pr)(Q)(M)(O)Aa...a` ASCII.

**Send Restart/Restart Confirmation Packets**
- `PRST(h1h2)(h3h4)` Sends a Restart packet.
- `PCRST` Restart Confirmation packet.

**Send Clear/Reset/Interrupt Confirmation Packets**
- `PUNCLEAR(h1h2)(h3h4)` Clear packet with data in hex.
- `PUNCLEAR(h1h2)(h3h4),(DAa...a)` Clear packet with data in ASCII.

**Send Data Packets**
- `PUND(Ps)(Pr)(Q)(M)(O)Hh...h` Hex.
- `PUND(Ps)(Pr)(Q)(M)(O)Aa...a` ASCII.

**Send Data Packets**
- `PUNCLEAR(h1h2)(h3h4)` Clear Confirmation packet.
- `PUNCLEAR(h1h2)(h3h4),DH` Data in hex.
- `h1h2 = cause code
  h3h4 = diagnostic code`

**Send Information**
- `FNS(Nr)(,Pr)(PACKET)` Packet mnemonic.
- `FNS(Nr)(,Pr)(PHh1h2...h)` Packet in hex.
- `FNS(Nr)(,Pr)(PAabcd...a)` Packet in ASCII.

**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` Displays text on the screen.
- `LPRINT` 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: Sets packet level ON.
- SPOF: Sets 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: frame window size, range 1 - 7
- SKx(T or R): packet window size, range 1 - 7
- T = Transmit window size
- R = Receive window size
- x = 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)

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: LCGN in hex
- Assigns a default LCGN for the next placed call.

Set Interface Leads
- SNP+: Sets interface lead nnn active (space).
- SNP-: Sets interface lead nnn inactive (mark).
- nnn is one of the signal numbers:
  - DCE: 106
  - DSR: 107
  - DCD: 109
  - DDCD: 122
  - RI: 125
  - DTE: 105
  - DTR: 108

Test Interface Leads
- IFnnn+dddd: Interface signal is active.
- IFnnn- dddd: Interface signal is inactive.

Set Timers
- ST'h1h2h3h4: Sets timer T'.
- ST"h1h2: Sets timer T''.
- STUnh1h2h3h4: Sets user-defined timer TU.

Set Counters
- SChn1h2: Sets counter n to hex value h1h2.
- SCh+: Increments counter n.
- SCh-: 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
- SunVPn1h2h3: 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.
- **SXAlXB** Logical Exclusive OR (XOR).

**Shift and Rotate**
- **SXADn** Shifts XA n times to the right.
- **SXAn** 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=1h1h2 dddd** XA equals h1h2.
- **IXA#XB dddd** XA is not equal to XB.
- **IXA#1h1h2 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>1h1h2 dddd** XA is greater than h1h2.
- **IXA(XB dddd** XA <- to XB.
- **IXA(h1h2 dddd** XA -> to h1h2.
- **IXA(XB dddd** XA <- to h1h2.
- **IXA(h1h2 dddd** XA -> to 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.
- **SXAXB** Inserts value of variable XA in the byte of the buffer indicated by the value contained in variable XB.
- **SXAh1h2** Inserts the value of XA in the byte of the buffer indicated by the 2-digit hex value h1h2.

**Message Buffer Length**
- **SXAl00** 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.
- **ISXXYY,XXYY,...** 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.
SESELSE 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 — Tests 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 — SEEnd 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 — Stops scenario execution.
### SITREX AUTOMATIC SIMULATOR COMMANDS

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Sets the Automatic X.25 Simulator to echo Called and Calling Addresses in Call Confirmation packets.</td>
</tr>
<tr>
<td>PP</td>
<td>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.</td>
</tr>
<tr>
<td>PS</td>
<td>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.</td>
</tr>
</tbody>
</table>

### SITREX DEFAULT PSEUDO–USERS

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>CONFIGURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Reserved for the Chameleon 32 Trace page (pink window).</td>
</tr>
<tr>
<td>02</td>
<td>Reserved for the Chameleon 32 Simulation page (blue window).</td>
</tr>
<tr>
<td>03</td>
<td>Traffic Generator</td>
</tr>
<tr>
<td>04</td>
<td>Echo Generator</td>
</tr>
<tr>
<td>05</td>
<td>Data Absorber</td>
</tr>
<tr>
<td>06</td>
<td>Second Traffic Generator</td>
</tr>
<tr>
<td>07</td>
<td>Third Traffic Generator</td>
</tr>
</tbody>
</table>
SITREX TRACE PAGE COMMANDS

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

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

- Transmitted
- Good CRC
- Frame Length
- Address
- I-Frame
- N(R)
- N(S)

Data in hex
D - Indicates data
Calling Address
Call Packet
GFI = Mod128
LGCN = 0
LCN = 07
/ = Packet Level

T* 030 01 I 6 5 /E007 DA Q D 000 M 000 48 45 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 is 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 \_\_\_\_ 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  \_\_\_\_\_\_ out of sequence and station not busy and no Tx.RNR requested and no prior Tx.REJ OR Tx.P/F set.
T17  \_\_\_\_\_\_ 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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>Runs program in background mode. progname &amp;</td>
</tr>
<tr>
<td>#</td>
<td>Programmer’s remark. #text</td>
</tr>
<tr>
<td>'</td>
<td>Echoes text to screen. 'text</td>
</tr>
<tr>
<td>batch</td>
<td>Executes a batch file batch filename</td>
</tr>
<tr>
<td>cat</td>
<td>Concatenates and prints files cat filename</td>
</tr>
<tr>
<td>cd</td>
<td>Changes current directory cd path</td>
</tr>
<tr>
<td>cp</td>
<td>Copies files into a directory cp oldfile newfile</td>
</tr>
<tr>
<td>ctags</td>
<td>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</td>
</tr>
<tr>
<td>dump</td>
<td>Prints files in hex to standard output. dump filename filename</td>
</tr>
<tr>
<td>exit</td>
<td>Exits C shell. Returns to main menu. exit</td>
</tr>
<tr>
<td>format</td>
<td>Formats a floppy disk. format</td>
</tr>
<tr>
<td>getenv</td>
<td>Displays environmental variable getenv name</td>
</tr>
<tr>
<td>help</td>
<td>Lists built-in shell commands help</td>
</tr>
<tr>
<td>jobs</td>
<td>Prints job control status jobs</td>
</tr>
<tr>
<td>kill</td>
<td>Kills a process that is running kill pid (pid=process ID)</td>
</tr>
<tr>
<td>ls</td>
<td>Lists files ls [-d] [-k] [-i] [-s] [spec]</td>
</tr>
<tr>
<td>man</td>
<td>Displays the named help file. man filename</td>
</tr>
<tr>
<td>mkdir</td>
<td>Creates a subdirectory mkdir dirname</td>
</tr>
<tr>
<td>mkres</td>
<td>Makes a program RAM resident mkres [−p] prog</td>
</tr>
<tr>
<td>more</td>
<td>Displays file or pipe output, one screen at a time more filename</td>
</tr>
<tr>
<td>mv</td>
<td>Moves a file mv file1 file2 Replace file2 with file1 mv file dir Move file to directory</td>
</tr>
<tr>
<td>psw</td>
<td>Prints current subdirectory psw</td>
</tr>
<tr>
<td>rm</td>
<td>Deletes one or more files rm filename filename...</td>
</tr>
<tr>
<td>rmdir</td>
<td>Deletes a subdirectory rmdir dirname</td>
</tr>
<tr>
<td>rmres</td>
<td>Removes a program from residency rmres pid (pid=process ID)</td>
</tr>
<tr>
<td>run</td>
<td>Runs a program as a separate process. run[−xxx] &lt;prog&gt; &amp;</td>
</tr>
<tr>
<td>setenv</td>
<td>Sets an environment variable setenv name ‘value’</td>
</tr>
<tr>
<td>shell</td>
<td>Opens a new page with the C shell. shell &amp;</td>
</tr>
<tr>
<td>size</td>
<td>Prints file size to standard output size filename filename...</td>
</tr>
<tr>
<td>time</td>
<td>Displays the system time. time</td>
</tr>
</tbody>
</table>
COMPILER COMMANDS

**cc** Compiles and links files
cc [-c] [flags] [file.c/file.o/...]
-c Compiles only, does not link files in source or object file

**mcc** Compiles C source files
mcc [-dname=value][-lpath] [file.c/file.o/...]
-dname Same as #define name in source
-lpath Searches path for include file.

MAKE UTILITY

**make** Executes commands in a makefile, causing related program files to be updated
make[opt] [-f mfile] [file]... [o file]
-opt One of the following options:
-1 Ignore error codes
-k Abandon work on current entry
-n No execute mode
-r Do not use built-in rules
-s Silent mode.
-t mfile Name of makefile to execute
-o 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 assembly file
-x Print hex value of instruction

LIBRARIAN COMMAND

**ar** Groups files into a single archive (object file libraries)
ar key [pos] ofile file
-key One of the following commands:
-o Output file.
-v Verbosed option.
-l 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.

EGREP COMMAND

egrep Searches files for user-defined patterns.
egrep.ttp [-C] [-f..] 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 line number of 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

<table>
<thead>
<tr>
<th>C LIBRARY FILENAME: libc.a</th>
<th>EXEC</th>
<th>EXECV</th>
<th>EXIT/EXIT</th>
<th>FCLOSE</th>
<th>FEAT</th>
<th>FERROR</th>
<th>FFLUSH</th>
<th>FGGETS</th>
<th>FGGETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Returns absolute value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># include &lt;stdio.h&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int abs(i)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>alloc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allocates RAM on the stack.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>char *alloca(size)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>atof</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Converts ASCII string to a floating-point number.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>double atof(nptr)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>atoi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int atoi(str)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>atol</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Converts ASCII string to a floating-point number.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>double atol(nptr)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bcmp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compares two blocks of memory.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int bcmp(block1, block2, len)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bcopy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copies a block of memory to another block.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int bcopy(source, destin, len)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bzero</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zeros a block of memory.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int bzero(block1, len)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calloc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allocates RAM for array of nelem elements of elsize size.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>char *calloc(nelem, elsize)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clearerr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resets error and EOF indicators to 0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#include &lt;stdio.h&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clearerr(stream)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>close</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Closes a file.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int close(filedes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Returns: 0 = Successful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1 = Error</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>creat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create file or overwrite existing file.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int creat(name, oflag)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Returns: 0 = Successful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1 = Error</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executes a file.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exec(name, arg0, arg1,...,argv,OL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execv</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executes a file.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execv(name, argv)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exit/exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminates a process. _exit returns without performing cleanup operations.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exit (status)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_exit (status)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fclose</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Writes all buffered data and closes stream.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#include &lt;stdio.h&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int fclose(stream)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ferror</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indicates when an I/O error occurs reading to/writing from the stream.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#include &lt;stdio.h&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int ferror(stream)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fflush</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Writes buffered data, but stream remains open.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#include &lt;stdio.h&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int fflush(stream)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fgetc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Same as getc, but is a true function.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>include &lt;stdio.h&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int fgetc(stream)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fggets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reads characters from stdin into array pointed to by s until EOF, n-1 characters are read.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#include &lt;stdio.h&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>char *fgets(s,n,stream)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>char 's;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int n;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE *stream;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### C Library Functions (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fileno</code></td>
<td>Returns the integer file descriptor.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int fileno(stream)</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td><code>fopen</code></td>
<td>Opens a file and associates a stream with it.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *fopen(file_name, type)</code></td>
</tr>
<tr>
<td></td>
<td><code>char *file_name;</code></td>
</tr>
<tr>
<td></td>
<td><code>char *type;</code></td>
</tr>
<tr>
<td></td>
<td><code>w = Create for writing</code></td>
</tr>
<tr>
<td></td>
<td><code>r = Open for reading</code></td>
</tr>
<tr>
<td></td>
<td><code>a = Append</code></td>
</tr>
<tr>
<td></td>
<td><code>r+ = Open for update</code></td>
</tr>
<tr>
<td></td>
<td><code>w+ = Create for update</code></td>
</tr>
<tr>
<td></td>
<td><code>a+ = Random open</code></td>
</tr>
<tr>
<td><code>fprintf</code></td>
<td>Places output to a named output stream.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int fprintf(stream, format [, arg] ..)</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td></td>
<td><code>char *format;</code></td>
</tr>
<tr>
<td><code>fputc</code></td>
<td>Similar to <code>putc</code>, but it is a true function.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int fputc(c, stream)</code></td>
</tr>
<tr>
<td></td>
<td><code>char c;</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td><code>fputs</code></td>
<td>Writes the string pointed to by <code>s</code> to stream.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int fputs(s, stream)</code></td>
</tr>
<tr>
<td></td>
<td><code>char *s;</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td><code>fread</code></td>
<td>Reads <code>nitems</code> of data from input stream at <code>ptr</code> and places in array.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int fread(ptr, size, nitems, stream)</code></td>
</tr>
<tr>
<td></td>
<td><code>char *ptr;</code></td>
</tr>
<tr>
<td></td>
<td><code>int size, nitems;</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td><code>free</code></td>
<td>Makes RAM at <code>ptr</code> available for allocation.</td>
</tr>
<tr>
<td></td>
<td><code>free(ptr)</code></td>
</tr>
<tr>
<td><code>freopen</code></td>
<td>Substitutes a file in place of the open stream.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *freopen(file_name, type, stream)</code></td>
</tr>
<tr>
<td></td>
<td><code>char *file_name;</code></td>
</tr>
<tr>
<td></td>
<td><code>char *type;</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td><code>fscanf</code></td>
<td>Reads from named input stream and converts formatted input.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int fscanf(stream, format [,pointer ..])</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td></td>
<td><code>char *format;</code></td>
</tr>
<tr>
<td><code>fseek</code></td>
<td>Sets position of next I/O operation on the stream.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int fseek(stream, offset, ptrname)</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td></td>
<td><code>long offset;</code></td>
</tr>
<tr>
<td></td>
<td><code>int ptrname;</code></td>
</tr>
<tr>
<td></td>
<td><code>Returns: 0 = Successful</code></td>
</tr>
<tr>
<td></td>
<td><code>Non-zero = Error</code></td>
</tr>
<tr>
<td><code>ftell</code></td>
<td>Returns offset of current byte relative to beginning of file.</td>
</tr>
<tr>
<td></td>
<td><code>long ftell(stream)</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td><code>fwrite</code></td>
<td>Appends <code>nitems</code> of data from array at <code>ptr</code> to an output stream.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int fwrite(ptr, size, nitems, stream)</code></td>
</tr>
<tr>
<td></td>
<td><code>char *ptr;</code></td>
</tr>
<tr>
<td></td>
<td><code>int size, nitems;</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td><code>getc</code></td>
<td>Returns the next byte in <code>stream</code> and advances pointer one byte.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int getc(stream)</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td><code>getchar</code></td>
<td>Returns the next character from stdin.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int getchar()</code></td>
</tr>
<tr>
<td><code>gets</code></td>
<td>Reads characters from stdin into array pointed to by <code>s</code> until EOL or new-line.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>char *gets(s)</code></td>
</tr>
<tr>
<td></td>
<td><code>char *s;</code></td>
</tr>
<tr>
<td><code>getw</code></td>
<td>Returns next word or integer from input stream.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;stdio.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int getw(stream)</code></td>
</tr>
<tr>
<td></td>
<td><code>FILE *stream;</code></td>
</tr>
<tr>
<td><code>is...</code></td>
<td>Character-coded integer values.</td>
</tr>
<tr>
<td></td>
<td><code>#include &lt;ctype.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>int isalpha c</code></td>
</tr>
<tr>
<td></td>
<td><code>Letter</code></td>
</tr>
<tr>
<td></td>
<td><code>int isupper c</code></td>
</tr>
<tr>
<td></td>
<td><code>Upper case letter</code></td>
</tr>
<tr>
<td></td>
<td><code>int islower c</code></td>
</tr>
<tr>
<td></td>
<td><code>Lower case letter</code></td>
</tr>
<tr>
<td></td>
<td><code>int isdigit c</code></td>
</tr>
<tr>
<td></td>
<td><code>Digit</code></td>
</tr>
<tr>
<td></td>
<td><code>int isalnum c</code></td>
</tr>
<tr>
<td></td>
<td><code>Alphanumeric</code></td>
</tr>
<tr>
<td></td>
<td><code>int isspace c</code></td>
</tr>
<tr>
<td></td>
<td><code>Space, tab, CR, newline, form feed</code></td>
</tr>
<tr>
<td></td>
<td><code>int ispunct c</code></td>
</tr>
<tr>
<td></td>
<td><code>Punctuation</code></td>
</tr>
<tr>
<td></td>
<td><code>int isprint c</code></td>
</tr>
<tr>
<td></td>
<td><code>Print chars: 040–0176</code></td>
</tr>
<tr>
<td></td>
<td><code>int isctrl c</code></td>
</tr>
<tr>
<td></td>
<td><code>Delete chars: 0177 or control chars &lt; 040</code></td>
</tr>
<tr>
<td></td>
<td><code>int isascii c</code></td>
</tr>
<tr>
<td></td>
<td><code>ASCII chars: &lt; 0200</code></td>
</tr>
<tr>
<td></td>
<td><code>int isxdigit c</code></td>
</tr>
<tr>
<td></td>
<td><code>Hex digit</code></td>
</tr>
<tr>
<td></td>
<td><code>int c;</code></td>
</tr>
<tr>
<td></td>
<td><code>Returns: 0 = False Non-zero=True</code></td>
</tr>
</tbody>
</table>

**Note:** The above table includes a variety of C library functions, each with its own specific use case. For example, `fopen` is used to open a file and `ftrace` for tracing purposes. Each function is accompanied by its prototype and a brief description of its operation. The code listings in C are also included, providing a practical example of their usage.
### C Library Functions (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>malloc</code></td>
<td>Returns pointer to a block of RAM of size or greater. Like malloc but accepts a long parameter.</td>
</tr>
<tr>
<td><code>longjmp</code></td>
<td>Restores the environment saved by <code>setjmp</code> in <code>env</code>.</td>
</tr>
<tr>
<td><code>realloc</code></td>
<td>RAM allocator which changes the size of the block pointed to by <code>ptr</code> to <code>size</code> bytes. Like realloc, but accepts a long parameter.</td>
</tr>
<tr>
<td><code>lseek</code></td>
<td>Moves the file pointer according to <code>whence</code>.</td>
</tr>
<tr>
<td><code>calloc</code></td>
<td>Returns pointer to a block of RAM (64K bytes or less) of <code>size</code> or greater.</td>
</tr>
<tr>
<td><code>open</code></td>
<td>Opens a file described for <code>fname</code> and sets flag to <code>oflag</code>.</td>
</tr>
<tr>
<td><code>perror</code></td>
<td>Writes an error message onto the standard stream.</td>
</tr>
<tr>
<td><code>printf</code></td>
<td>Places output on stdout.</td>
</tr>
<tr>
<td><code>putc</code></td>
<td>Writes character <code>c</code> to the output stream at current pointer position.</td>
</tr>
<tr>
<td><code>putchar</code></td>
<td>Equivalent to <code>putc(c, stdout)</code>.</td>
</tr>
<tr>
<td><code>puts</code></td>
<td>Writes the string pointed to by <code>s</code> to stdout.</td>
</tr>
<tr>
<td><code>putw</code></td>
<td>Writes the word (integer) <code>w</code> to the output stream at current pointer position.</td>
</tr>
<tr>
<td><code>qsort</code></td>
<td>Quicksort algorithm.</td>
</tr>
<tr>
<td><code>rand</code></td>
<td>Generates a random number.</td>
</tr>
<tr>
<td><code>read</code></td>
<td>Reads <code>nbyte</code> bytes from the file into the buffer pointed to by <code>buf</code>.</td>
</tr>
<tr>
<td><code>rewind</code></td>
<td>Equivallent to <code>fseek(stream,OK,0)</code>, but no value is returned.</td>
</tr>
<tr>
<td><code>scanf</code></td>
<td>Reads from stdin and converts formatted input.</td>
</tr>
<tr>
<td><code>setbuf</code></td>
<td>Assigns buffer pointed to by <code>buf</code> to a stream.</td>
</tr>
<tr>
<td><code>setjmp</code></td>
<td>Non-local goto which saves its stack environment in <code>env</code> for use by <code>longjmp</code>.</td>
</tr>
</tbody>
</table>

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int malloc(size);
int calloc(ptr, size);
int realloc(ptr, size);
int putc(c, stream);
int putchar(c);
int puts(s);
int putw(w, stream);
FILE *stream;
FILE *stdout;
```
C LIBRARY FUNCTIONS (CONTINUED)

**sprintf**
Places output in consecutive bytes starting at s.

```c
#include<stdio.h>
int sprintf(s,format[,arg1...])
```

**srand**
Resets the random number generator to a new starting point.

```c
#include <stdio.h>
srand(seed)
```

**sscanf**
Reads from character string a and converts formatted input.

```c
#include <stdio.h>
int sscanf(s,format[,pointer1...])
```

**strtol**
Converts string to long integer.

```c
#include <stdio.h>
long strtol(str,ptr,base)
```

**tolower**
Returns argument with all but the 7 low order bits set to zero.

```c
#include <ctype.h>
int toascii (c)
```

**toupper**
Converts characters to upper case.

```c
#include <ctype.h>
int toupper (c)
```

**ungetc**
Inserts character c into buffer.

```c
#include <stdio.h>
int ungetc(c,stream)
```

**unlink**
Removes a directory entry.

```c
#include <stdio.h>
int unlink(path)
```

**write**
Writes nbyte bytes from buffer pointed to by buf to the file.

```c
#include <stdio.h>
int write(fildes,buf,nbyte)
```

### STRING OPERATIONS

```c
#include <string.h>
```

- **char *strcat(s1,s2)** Appends s2 to end of s1.
- **char *strncpy(s1,s2,n)** Copies n characters from s2 to end of s1.
- **int strcmp(s1,s2)** Compares n characters of s2 to s1. Returns:
  0 if s1 == s2
  >0 if s1 > s2
  <0 if s1 < s2
- **int strncmp(s1,s2,n)** Compares n characters of s2 to s1.
- **int strlen(s)** Returns length of s.
- **int index(s,c)** Move pointer to first c in s.
- **int rindex(s,c)** Move pointer to last c in s.
- **int strcmp(s1,s2)** Compares s1 and s2.

**AUX. SERIAL PORT 2 FUNCTIONS**

**initporta** Initializes Aux Serial Port 2.

```c
#include "paval.h"
int initporta (stopbit, bitchar, bitrate, parity)
```

- **stopbit**
  - ST1: 1 stop bit
  - ST15: 1.5 stop bit
  - ST20: 2 stop bit
- **bitchar**
  - O8: 5 data bits
  - O86: 6 data bits
  - O87: 7 data bits
  - O88: 8 data bits
- **bitrate**
  - F110: 110 bps
  - F300: 300 bps
  - F500: 450 bps
  - F1200: 2400 bps
  - F1500: 2400 bps
  - F1600: 1200 bps
  - F19200: 9600 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.

```c
#include "paval.h"
int sndpa (ptr, nb, timeout)
```

- **指针**
- **nb**
- **timeout**

Returns:
- nb: Number of bytes transmitted
- 0: Timeout
- -1: Parameter error

**recpa** Receives data using Aux Serial Port 2.

```c
#include "paval.h"
int recpa (ptr, timeout)
```

- **指针**
- **timeout**

Returns:
- nb: Number of bytes received
- 0: Timeout
- -1: Parameter error

**rstdrv** Flashes the driver reception buffer.

```c
#include "paval.h"
int rstdrv()
```
C LIBRARY FUNCTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>assignleds</td>
<td>Creates or changes LEDs.</td>
</tr>
<tr>
<td>closeform</td>
<td>Changes from form to window mode.</td>
</tr>
<tr>
<td>closevt</td>
<td>Releases the virtual terminal.</td>
</tr>
<tr>
<td>disablecur</td>
<td>Causes the cursor to be invisible.</td>
</tr>
<tr>
<td>enablecur</td>
<td>Causes the cursor to be visible.</td>
</tr>
<tr>
<td>getch</td>
<td>Gets a character from stdin.</td>
</tr>
<tr>
<td>getcwd</td>
<td>Waits for a character from stdin.</td>
</tr>
<tr>
<td>openform</td>
<td>Opens a form.</td>
</tr>
<tr>
<td>openvt</td>
<td>Assigns a virtual terminal.</td>
</tr>
<tr>
<td>prndata</td>
<td>Sends data to the printer.</td>
</tr>
<tr>
<td>putvt</td>
<td>Displays a string on a virtual terminal.</td>
</tr>
<tr>
<td>seiprn</td>
<td>Selects the parameters for a printer.</td>
</tr>
</tbody>
</table>

MATH LIBRARY: libm.a

```c
#include <math.h>

double log(x) // Base 10 logarithm.
double log10(x) // Base 10 logarithm.
double log2(x); // Base 2 logarithm.

double exp(x) // Base e exponential.
double exp10(x) // Base 10 exponential.

double sin(x) // Transcendental functions.
double cos(x)

double tan(x)

double asin(x) // Inverse transcendental functions.
double acos(x)
double atan(x)

double sqrt(x) // Absolute value (?x?).
double sqrt(x)

double power(x, y) // xy (equivalent to exp2(x*log2(y))

double power2(x, k) // Fast floating point multiplication by 2k.

double lngamma(x) // Natural logarithm of the gamma function 0<x<5.1 X10305.

double fac(k) // kl, where 0<k<170

double matinv(a, c, n) // Matrix double matinv(a,c,n)

double "a"; // double "a";

MS–DOS File Compatible Functions

Fmkdir | Creates a directory. |
Fmdir | Removes a directory. |
Fsearch | Searches for a file or directory. |

MS–DOS File Compatible Functions

Fmkdir(dname)
char *dname; (including path)

Fmkdir(dname)

Frmdir(dname)
char *dname; (including path)

Fsearch(name, option, rec)
char *name; (file/dir name and path)
# VI Commands

## Key

<table>
<thead>
<tr>
<th>n</th>
<th>Number</th>
<th>x</th>
<th>Single upper or lower case character</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt</td>
<td>Option</td>
<td>CR</td>
<td>RETURN key (carriage return)</td>
</tr>
<tr>
<td>^</td>
<td>CTRL key</td>
<td>pat</td>
<td>Text and/or pattern matching characters</td>
</tr>
</tbody>
</table>

## 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.
- **nx**: Delete n characters.
- **X**: Delete character before cursor.
- **dw**: Delete word.
- **ndw**: Delete n words.
- **dd**: Delete line.
- **ndd**: Delete n lines.
- **dx**: Delete to x in a line.
- **D**: Delete rest of line.
- **d?patCR**: Delete up to pat.
- **d!patCR**: Delete up 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

- **mx**: 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.
- **ncw**: Change n words until ESC.
- **cc**: Change line until ESC.
- **ncc**: Change n lines until ESC.
- **ctx**: Change to x until ESC.
- **C**: Change rest of line until ESC.
- **c?patCR**: Change back to pat.
- **c?patCR**: Change up to pat.

## Go to (File) Commands

- **G**: Go to last line of file.
- **rG**: 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 nth line after pat.
- **?pat?−n**: Go to nth 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**: Edit 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.
### MOVE TO (Screen) COMMANDS

- Move to 1st character, next line.
- Move to 1st char. previous line.
- Move to next line, same column.
- Move to previous line.
- Move to next line.
- Move to previous line.
- Move to right.
- Move to left.
- Move to end of line.

### SCREEN ADJUSTMENT COMMANDS

- Clear and redraw screen.
- Redraw current line at top.
- Redraw current line at center.
- Redraw window with pat line at bottom.
- Redraw window with n lines.
- Scroll window down 1 line.
- Scroll window up 1 line.
- Scroll forward 1 screen.
- Scroll backward 1 screen.
- Scroll down half a screen.
- Scroll up half a screen.

### SEARCH PATTERNS COMMANDS

- Beginning of line.
- End of line.
- Any character.
- 0 or more repetitions of character.
- Match any character from A–Z.
- Match a, b or c.
- Match any char. except a, b, c.
- Escape character for literal \ / $.
- Beginning of word.
- End of word.

### SUBSTITUTION COMMANDS

- Substitute text for character.
- Substitute line.
- Substitute first X with Y.
- Change all occurrences.
- Confirm each change.
- Print each change.

### UNDO COMMANDS

- Undo last operation.
- Restore current line.
- Retrieve n'th last delete.
- Scan previous n deletes.

### WORD AND LINE COMMANDS

- Move forward 1 word.
- Move forward 1 word, including punctuation.
- Move back 1 word.
- Move back 1 word, including punctuation.
- Move to end of word.
- Move to end of word, including punctuation.
- Find x forward (to the right).
- Reverse last f.

### YANK AND PULL COMMANDS

- Yank line to buffer.
- Yank n lines to buffer.
- Put lines back below cursor.
- Put lines back above cursor.
- Yank to buffer x.
- Put from buffer x.

### vi SOFTKEYS

**FILE Softkeys**

- Open
- Save
- Quit
- Save/Quit
- Revert

**EDIT Softkeys**

- Insert
- Append
- Del chr
- Cut
- Copy

**F1**

- F6 Read
- F7 Set
- F8 Next
- F9 Rewind
- F10 EDIT Softkeys
These functions are available in all protocol libraries.

**FLUSH**
Clears all outstanding frames in the reception buffer.

```c
int flush()

Returns: 3 Receive buffer overflow
```

**GETPHY**
Indicates physical interface setting.

```c
int getphy()

Returns 2-byte integer as shown below.
```

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Bit 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 105 108 140 141 104 103 114 115</td>
<td></td>
</tr>
<tr>
<td>Pin 4  20  3  2  15  17</td>
<td></td>
</tr>
<tr>
<td>Sig RTS DTR SQ RD TD SCT SCR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte 1</th>
<th>Bit 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 106 107 109 122 125 142</td>
<td></td>
</tr>
<tr>
<td>Pin 5  6  8  12  22</td>
<td></td>
</tr>
<tr>
<td>Sig CTS DSR CD SDCC RI</td>
<td></td>
</tr>
</tbody>
</table>

**GETPORT**
Identifies which port is communicating with the library.

```c
int getport()

Returns: 0 Port A selected
1 Port B selected
```

**GETTIME**
Gets the number of milliseconds since the system was started.

```c
#include <mtoS-ux.h>
int gettime(
unsigned char *msbfr;

unsigned char *msbfr;
```

**INITTIME**
Initializes the .01 and 1 second timers.

```c
Use init1 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.

```c
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.

```c
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.

```c
setphy()
```

<table>
<thead>
<tr>
<th>DCE Bit 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 106 107 109 122 125 142</td>
</tr>
<tr>
<td>Pin 5 6 8 12 22</td>
</tr>
<tr>
<td>Sig CTS DSR CD SDCC RI</td>
</tr>
</tbody>
</table>

## C PROTOCOL LIBRARY COMMON FUNCTIONS (continued)

**SETPHY**

```c
DCE Bit 7 6 5 4 3 2 1 0
Pin 105 108 140 141 104 103 114 115
Pin 4 20
Sig RTS DTR SQ
```

**SETPORT**
Selects Port A or Port B for library.

```c
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.

```c
setter(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.

```c
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
-202 FEP Parameter port
-203 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**

**FILENAME:** libhdlc.a

**INITP1**
Initializes P1 and loads software.

```c
int initp1(type1, type2, encode, bitrate)
int type1;  // 0=DCE, 1=DE
int type2;  // 0=Network, 1=Subscriber
int encode; // 0=NRZ, 1=NRZ
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.

```c
char receive (packet)
char *packet;
```

Returns:
- 0=Successful
- 1=Link not established
- 2=initp1 not performed

**SET_N1**
Sets N1 (maximum packet size).

```c
int set_n1(val)
int val;  // Range: 1 - 512
```

Returns:
- 0=Successful
- -1=Invalid value

**SET_N2**
Sets N2 (retransmissions).

```c
int set_n2(val)
int val;  // Range: 1 - 255
```

Returns:
- 0=Successful
- -1=Invalid value

**SET_T1**
Sets T1 timer.

```c
int set_t1(val)
int val;  // Range: 1 - 255
```

Returns:
- 0=Successful
- -1=Invalid value

**SET_WINDOW**
Sets window size.

```c
int set_window(val)
int val;  // Range: 1 - 7
```

Returns:
- 0=Successful
- -1=Invalid value

**SLOF**
Disconnects link.

```c
int slof()
```

**SLON**
Establishes link by sending a SABM.

```c
int slon()
```

**STATUS**
Indicates status of frame level.

```c
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 frame.

```c
int 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.

```c
int discard()
```

Returns:
- 0 Frame discarded; no frame in buffer.
- <0 Standard error codes

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

```c
int initp1(type, encode, bitrate, flag)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>0=DCE 1=DTE 2=ISDN</td>
</tr>
<tr>
<td><code>encode</code></td>
<td>0=NRZ 1=NRZI</td>
</tr>
<tr>
<td><code>bitrate</code></td>
<td>50 - 64000</td>
</tr>
<tr>
<td><code>flag</code></td>
<td>0=FF 1=7E</td>
</tr>
</tbody>
</table>

Returns:
- 0 Successful
- 1=Invalid parameter(s)
- 2=P1 program not loaded

**GET_NXLEN** Returns the length of next frame from FEP.

```c
int get_nxlen()
```

Returns:
- 0 No new frame
- >0 length of next frame
- <0 Standard error codes

**GET_NXSTAT** Gives status of next frame.

```c
int get_nxstat()
```

Returns:
- 0 No new frame
- 1 Frame ok
- 2 Parity error in frame
- 3 Abort sequence in frame
- <0 Standard 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.

```c
int initp_8k1(type, encode, bitrate, flag)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>0=DCE</td>
</tr>
<tr>
<td></td>
<td>1=DTE</td>
</tr>
<tr>
<td><code>encode</code></td>
<td>0=NRZ</td>
</tr>
<tr>
<td></td>
<td>1=NRZI</td>
</tr>
<tr>
<td><code>bitrate</code></td>
<td>50 - 64000</td>
</tr>
<tr>
<td><code>flag</code></td>
<td>0=FF 1=7E</td>
</tr>
</tbody>
</table>

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.

```c
char receive(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.

```c
int selflg(flag)
```

Returns:
- 0=Successful
- 1=Invalid parameter(s)
- 2=P1 program not loaded

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

```c
int transmit(mode, frame, length)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mode</code></td>
<td>0=Good CRC 1=Bad CRC</td>
</tr>
<tr>
<td><code>bitrate</code></td>
<td>2=Abort sequence</td>
</tr>
</tbody>
</table>

Returns:
- 0=Successful
- 1=P1 busy
- 2=initp1 not performed
- 3=Parameter error
- 4=Buffer overflow

**TREADY** Returns status of P1 transmitter.

```c
int tready()
```

Returns:
- 0=Transmitter ready
- 1=Transmitter not ready
- 2=initp1 not performed
- 3=Overflow
# LAPD C Library Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Returns/Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET_MOD</td>
<td>Returns the current modulus.</td>
<td><code>int get_mod()</code> Returns: 0=Mod8, 1=Mod128</td>
</tr>
<tr>
<td>GET_RNTETI</td>
<td>Returns value of user-defined receive TEI.</td>
<td><code>int get_rntei(val)</code> Returns: val; Range: 0 - 2</td>
</tr>
<tr>
<td>GET_RSAPI</td>
<td>Returns value of user-defined SAPI.</td>
<td><code>int get_rsapi(val)</code> Returns: val; Range: 0 - 2</td>
</tr>
<tr>
<td>GET_SCONFIG</td>
<td>Returns status configuration byte.</td>
<td><code>int get_sconfig</code> Returns status configuration byte (see manual for interpretation).</td>
</tr>
<tr>
<td>GET_SIM</td>
<td>Returns the side being simulated.</td>
<td><code>int get_sim()</code> Returns: 0=Network, 1=Subscriber</td>
</tr>
<tr>
<td>INITPI</td>
<td>Initializes P1 and loads software.</td>
<td><code>int initpi(interface,station,encode,bitrate)</code> Returns: 0=Successful, 1=Error</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>Receives an I-frame from P1 and places it at address in rloc.</td>
<td><code>extern int rxlen</code> <code>int receive(rloc)</code> Returns: 0=Successful or no frame waiting, 2=initp1 not performed, 4=P1 busy</td>
</tr>
<tr>
<td>RESTARTSIM</td>
<td>Restarts P1 simulation.</td>
<td><code>int restartsim()</code> Returns: 0=Successful, 1=Time out</td>
</tr>
<tr>
<td>SETFLG</td>
<td>Changes the idle fill pattern.</td>
<td><code>int setflg(flag)</code> Returns: 0=Fill with FF, 1=Fill with 7E</td>
</tr>
<tr>
<td>SET_BIT_RATE</td>
<td>Sets the bit rate.</td>
<td><code>long rate</code> Returns: 0=Successful, 1=Error</td>
</tr>
<tr>
<td>SET_MOD</td>
<td>Sets the modulus to mod8 or mod128.</td>
<td><code>int set_mod(val)</code> Returns: 0=Successful, 1=Mod128, 1=Mod128</td>
</tr>
<tr>
<td>S_N200</td>
<td>Sets N2 (retransmissions).</td>
<td><code>int s_n200(val)</code> Returns: 0=Successful, 1=Time out</td>
</tr>
<tr>
<td>S_N201</td>
<td>Sets N1 (maximum packet size).</td>
<td><code>int s_n201(val)</code> Returns: 0=Successful, 1=Time out</td>
</tr>
</tbody>
</table>
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_rsaapi(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.
   int set_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_t200(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 or 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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Parameters</th>
<th>Return Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STATUS</strong></td>
<td>Indicates status of frame level.</td>
<td>int status()</td>
<td>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</td>
</tr>
<tr>
<td><strong>STOPSIM</strong></td>
<td>Stops P1 simulation.</td>
<td>int stopsim()</td>
<td>0=Successful, 1=Time out</td>
</tr>
<tr>
<td><strong>TRANS</strong></td>
<td>Transmits a specified type of frame.</td>
<td>int trans(stat,frame,len)</td>
<td>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</td>
</tr>
<tr>
<td><strong>TRANSMIT</strong></td>
<td>Transmits number of bytes in length, starting at address in packet.</td>
<td>int transmit(packet, length)</td>
<td>Calls and returns the value returned by: trans(IFrame, packet, length)</td>
</tr>
<tr>
<td><strong>TRUI</strong></td>
<td>Transmits an unnumbered I-frame.</td>
<td>int trui(xloc,xlen)</td>
<td>char *xloc; Location of data, int xlen; Length of data field</td>
</tr>
<tr>
<td><strong>TRXCN</strong></td>
<td>Transmits an XID command frame without an I-field.</td>
<td>int trxcni()</td>
<td>Returns: 0=Successful, 1=Time out</td>
</tr>
<tr>
<td><strong>TRXIDC</strong></td>
<td>Transmits an XID command frame with an I-field.</td>
<td>int trxidc(xloc,xlen)</td>
<td>char *xloc; Location in memory, int xlen; Length</td>
</tr>
<tr>
<td><strong>TRXIDR</strong></td>
<td>Transmits an XID response frame with an I-field.</td>
<td>int trxidr(xloc,xlen)</td>
<td>char *xloc; Location in memory, int xlen; Length</td>
</tr>
<tr>
<td><strong>TRXRNI</strong></td>
<td>Transmits an XID response frame without an I-field.</td>
<td>int trxrni()</td>
<td>Returns: 0=Successful, 1=Time out</td>
</tr>
</tbody>
</table>
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=NRZ
unsigned long bitrate; 50 - 64000
Returns:
0=Successful
-1=Invalid parameter(s)
-2=P1 program not loaded

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() Returns:
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

RECEIVE Receives a frame from P1 and places it at address in packet.
char receive(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 *packet; 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

TRANSIT Transmits I-frame with I-field of length, starting at address in packet.
int transmit(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 trnsi(char *packet, int length)
Returns:
0=Successful
1=P1 busy
2=initp1 not performed
3=Link not established
4=Length error (if length > 510)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRSIFR</strong></td>
<td>Transmits a sequenced I-frame with I-field of length, starting at address in packet.</td>
<td><code>int trsifr(packet, length)</code>&lt;br&gt; <code>char *packet;</code>&lt;br&gt; <code>int length;</code>&lt;br&gt; Returns:&lt;br&gt; 0=Successful&lt;br&gt; 1=P1 busy&lt;br&gt; 2=initp1 not performed&lt;br&gt; 3=Link not established&lt;br&gt; 4=Length error (if length &gt; 510)</td>
</tr>
<tr>
<td><strong>TRUI</strong></td>
<td>Transmits an unnumbered I-frame with I-field of length, starting at address in packet.</td>
<td><code>int trui(packet, length)</code>&lt;br&gt; <code>char *packet;</code>&lt;br&gt; <code>int length;</code>&lt;br&gt; Returns:&lt;br&gt; 0=Successful&lt;br&gt; 1=P1 busy&lt;br&gt; 2=initp1 not performed&lt;br&gt; 3=Link not established&lt;br&gt; 4=Length error (if length &gt; 510)</td>
</tr>
<tr>
<td><strong>TRTST</strong></td>
<td>Transmits a test frame with I-field of length, starting at address in packet.</td>
<td><code>int trtst(packet, length)</code>&lt;br&gt; <code>char *packet;</code>&lt;br&gt; <code>int length;</code>&lt;br&gt; Returns:&lt;br&gt; 0=Successful&lt;br&gt; 1=P1 busy&lt;br&gt; 2=initp1 not performed&lt;br&gt; 3=Link not established&lt;br&gt; 4=Illegal frame (if secondary)&lt;br&gt; 5=Not a primary station</td>
</tr>
<tr>
<td><strong>XID</strong></td>
<td>Transmits an XID frame containing the data in the externally available character array ident[].</td>
<td><code>extern char ident[ ];</code>&lt;br&gt; /* 6 bytes */&lt;br&gt; <code>char xid();</code>&lt;br&gt; Returns:&lt;br&gt; 0=Successful&lt;br&gt; 1=P1 not initialized&lt;br&gt; 2=P1 fails to respond&lt;br&gt; 3=Not in normal response mode&lt;br&gt; 4=Illegal frame (if secondary)</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>resblock [0]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Successful</td>
</tr>
<tr>
<td>01</td>
<td>Hardware has already been set up</td>
</tr>
<tr>
<td>02</td>
<td>Requested function is not available for this configuration</td>
</tr>
<tr>
<td>03</td>
<td>Requested channel is invalid (for B1, B2 and D)</td>
</tr>
<tr>
<td>04</td>
<td>Requested function is not available for this channel</td>
</tr>
<tr>
<td>05</td>
<td>Invalid command or request</td>
</tr>
<tr>
<td>10</td>
<td>Basic Rate Interface board is not installed</td>
</tr>
</tbody>
</table>

Setup     
cmdblock[0] = 1 (Board 0) or cmdblock[0] = 101 (Board 1)
cmdblock[1]  
<table>
<thead>
<tr>
<th>mode</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Monitor</td>
</tr>
<tr>
<td>2</td>
<td>Simulate NT</td>
</tr>
<tr>
<td>3</td>
<td>Simulate TE</td>
</tr>
</tbody>
</table>

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]  
<table>
<thead>
<tr>
<th>mode</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>If request conflicts with current setup, do not override.</td>
</tr>
<tr>
<td>1</td>
<td>If request conflicts with current setup, override.</td>
</tr>
</tbody>
</table>

cmdblock[2]  
<table>
<thead>
<tr>
<th>channel</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B1 channel</td>
</tr>
<tr>
<td>2</td>
<td>B2 channel</td>
</tr>
<tr>
<td>3</td>
<td>D channel</td>
</tr>
</tbody>
</table>

cmdblock[3]  
<table>
<thead>
<tr>
<th>selection</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>System</td>
</tr>
<tr>
<td>2</td>
<td>Milliwatt</td>
</tr>
<tr>
<td>3</td>
<td>Codec</td>
</tr>
<tr>
<td>4</td>
<td>External interface</td>
</tr>
<tr>
<td>5</td>
<td>Idle</td>
</tr>
</tbody>
</table>

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]  
<table>
<thead>
<tr>
<th>For NT</th>
<th>Deactivate request</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Send info–2</td>
</tr>
<tr>
<td>3</td>
<td>Send info–4</td>
</tr>
<tr>
<td>4</td>
<td>Activate NT</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>Send single pulses</td>
</tr>
<tr>
<td>7</td>
<td>Send continuous pulses</td>
</tr>
<tr>
<td>8</td>
<td>Send info–2, test loop 2</td>
</tr>
<tr>
<td>9</td>
<td>Send info–4, test loop 2</td>
</tr>
</tbody>
</table>

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  |
Get Status  \texttt{cmdblock[0] = 6 (Board 0) or cmdblock[0] = 106 (Board 1)}
\textbf{Argument}  None
\texttt{resblock[1]}  Control byte received from PEB 2080.
\texttt{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. Deactivation complete
8. Undefined
\texttt{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:
\texttt{resblock[1]}  Control byte received from PEB 2080.
\texttt{resblock[2]}  Same as \texttt{resblock[2]} from NT
\texttt{resblock[3]}  Same as \texttt{resblock[2]} from TE

Select Trace Option
\texttt{cmdblock[0] = 9 (Board 0 only)}
\texttt{cmdblock[1]}  0  Turns off the trace
1  Command/result display
2  Detailed trace

NT Power  \texttt{cmdblock[0] = 10}
\texttt{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

\texttt{Bas_version}  This function returns the current version of the BRI library.
\texttt{char *Bas_version()}
2B1Q U-INTERFACE C LIBRARY FUNCTIONS

FILENAME: Jibu.a

```c
int SetU(cmdblock, resblock);
char cmdblock[];
char resblock[];
```

The error codes for resblock [0] for all U-Interface Library commands are listed below.

<table>
<thead>
<tr>
<th>resblock [0]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Successful</td>
</tr>
<tr>
<td>01</td>
<td>Invalid command</td>
</tr>
<tr>
<td>02</td>
<td>Invalid command parameters</td>
</tr>
<tr>
<td>03</td>
<td>Requested board is not responding</td>
</tr>
<tr>
<td>04</td>
<td>U-board physical error</td>
</tr>
<tr>
<td>05</td>
<td>U-board interface is not initialized</td>
</tr>
<tr>
<td>06</td>
<td>Requested board is not installed</td>
</tr>
</tbody>
</table>

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  
    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 = Normal Set
```

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

```
cmdblock[0] = 6 (Board 0) or cmdblock[0] = 106 (Board 1)
cmdblock[1] = Xcvr specifier 0 = NT Xcvr
                   1 = LT Xcvr
```

Get Transceiver

```
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.
```

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
```

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
```

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
```
2B1Q U-INTERFACE C LIBRARY FUNCTIONS (continued)

<table>
<thead>
<tr>
<th>resblock[2]</th>
<th>EOC address, EOC DM bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>resblock[3]</td>
<td>EOC information</td>
</tr>
<tr>
<td>resblock[4]</td>
<td>EOC address, EOC DM bit</td>
</tr>
<tr>
<td>resblock[5]</td>
<td>EOC information</td>
</tr>
<tr>
<td>resblock[6]</td>
<td>M4 information</td>
</tr>
<tr>
<td>resblock[7]</td>
<td>M5/M6 information</td>
</tr>
</tbody>
</table>

**EOC Processing**

<table>
<thead>
<tr>
<th>cmdblock[0]</th>
<th>13 (Board 0) or cmdblock[0] = 113 (Board 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmdblock[1]</td>
<td>Xcvr specifier 0 = NT Xcvr</td>
</tr>
<tr>
<td></td>
<td>1 = LT Xcvr</td>
</tr>
<tr>
<td>cmdblock[2]</td>
<td>Automatic processing mode</td>
</tr>
<tr>
<td></td>
<td>1 = No action</td>
</tr>
<tr>
<td></td>
<td>2 = Operate 2B + D Loopback</td>
</tr>
<tr>
<td></td>
<td>3 = Operate B1 Loopback</td>
</tr>
<tr>
<td></td>
<td>4 = Operate B2 Loopback</td>
</tr>
<tr>
<td></td>
<td>5 = Send corrupted CRC</td>
</tr>
<tr>
<td></td>
<td>6 = Return to Normal</td>
</tr>
</tbody>
</table>

**EOC Mode Control**

<table>
<thead>
<tr>
<th>cmdblock[0]</th>
<th>14 (Board 0) or cmdblock[0] = 114 (Board 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmdblock[1]</td>
<td>Xcvr specifier 0 = NT Xcvr</td>
</tr>
<tr>
<td></td>
<td>1 = LT Xcvr</td>
</tr>
<tr>
<td>cmdblock[2]</td>
<td>EOC Reception mode</td>
</tr>
<tr>
<td></td>
<td>1 = No action</td>
</tr>
<tr>
<td></td>
<td>2 = Handle every EOC</td>
</tr>
<tr>
<td></td>
<td>3 = Handle EOC passing trinal checks</td>
</tr>
<tr>
<td></td>
<td>4 = Handle EOC passing trinal checks with automatic EOC processing</td>
</tr>
</tbody>
</table>

**M4 Mode Control**

<table>
<thead>
<tr>
<th>cmdblock[0]</th>
<th>15 (Board 0) or cmdblock[0] = 115 (Board 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmdblock[1]</td>
<td>Xcvr specifier 0 = NT Xcvr</td>
</tr>
<tr>
<td></td>
<td>1 = LT Xcvr</td>
</tr>
<tr>
<td>cmdblock[2]</td>
<td>M4 Reception mode</td>
</tr>
<tr>
<td></td>
<td>0 = No action</td>
</tr>
<tr>
<td></td>
<td>1 = Handle dual–consecutive M4 with verified act/dea</td>
</tr>
<tr>
<td></td>
<td>2 = Handle dual–consecutive M4</td>
</tr>
<tr>
<td></td>
<td>3 = Handle Delta M4</td>
</tr>
<tr>
<td></td>
<td>4 = Handle every M4</td>
</tr>
</tbody>
</table>

**M5/6 Mode Control**

<table>
<thead>
<tr>
<th>cmdblock[0]</th>
<th>16 (Board 0) or cmdblock[0] = 116 (Board 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmdblock[1]</td>
<td>Xcvr specifier 0 = NT Xcvr</td>
</tr>
<tr>
<td></td>
<td>1 = LT Xcvr</td>
</tr>
<tr>
<td>cmdblock[2]</td>
<td>M4 Reception mode</td>
</tr>
<tr>
<td></td>
<td>0 = No action</td>
</tr>
<tr>
<td></td>
<td>1 = Handle dual–consecutive M5/6</td>
</tr>
<tr>
<td></td>
<td>3 = Handle Delta M5/6</td>
</tr>
<tr>
<td></td>
<td>4 = Handle every M5/6</td>
</tr>
</tbody>
</table>

**Shutdown**

| cmdblock[0] | 30 (Board 0) or cmdblock[0] = 130 (Board 1) |

81
**BSC C LIBRARY FUNCTIONS**

**BSC LIBRARY FILENAME:** libbsc.a

**IDLE_MODE** Specifies the character to be transmitted while the line is idle.

```c
#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.

```c
int initp1(type, encode, bitrate, crc, data)
int type; 0=OCE 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 die;
    unsigned char stx;
    unsigned char etx;
    unsigned char soh;
    unsigned char ebm;
    unsigned char erq;
};
unsigned long bitrate; 50 - 64000
char crc; 0=CRC16
1=CCITT-CRC
```

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

```c
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.

```c
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.

```c
int transmit(mode, frame, length)
int length;
char mode;
Returns:
0=Successful
1=P1 busy
2=initp1 not performed
3=Parameter error
4=Buffer overflow
```

---

**Diagram: Bit 7 to 0**
- **Start Framing Character**: 0=SOF 1=STX
- **End Framing Character**: 00=EOT 01=ETB 10=ETX 11=Illegal
- **Transparent Text Enable**: 0=Normal Text 1=Transparent Text
- **Transparent Mode**: 0=Transparent mode 0 1=Transparent mode 1
- **Text Mode**: 0=Control Mode 1=Text Mode
- **Block Check Character**: 0=Good BCC 1=Bad BCC
- **Reserved (must be 1)**
SetPrimary

int SetPrimary(cmdbuflock, resblock);
int cmdbuflock[14];
int resblock[14];

The error codes for resblock[0] for all Primary Rate Interface Library commands are:

<table>
<thead>
<tr>
<th>resblock[0]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful</td>
</tr>
<tr>
<td>1</td>
<td>Primary Rate Interface board is not installed</td>
</tr>
<tr>
<td>2</td>
<td>Setup already done</td>
</tr>
<tr>
<td>3</td>
<td>Invalid channel number/time slot</td>
</tr>
<tr>
<td>4</td>
<td>Selection already in use</td>
</tr>
<tr>
<td>5</td>
<td>Channel already assigned</td>
</tr>
<tr>
<td>10</td>
<td>Command not implemented</td>
</tr>
</tbody>
</table>

Setup

cmdbuflock[0] = 1 (Board A) or cmdbuflock[0] = 101 (Board B)

<table>
<thead>
<tr>
<th>cmdbuflock[1]</th>
<th>mode</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Monitor</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Simulate</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cmdbuflock[2]</th>
<th>framing</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ESF</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SL96</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CEPT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cmdbuflock[3]</th>
<th>idle data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8 bit value</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cmdbuflock[4]</th>
<th>idle signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 or 4 bit value</td>
<td></td>
</tr>
</tbody>
</table>

*cmdbuflock[5] | DS0x receive | Channel/time slot |
*cmdbuflock[6] | Codec receive | Channel/time slot |
*cmdbuflock[7] | DS0y receiver/transmitter | Channel/time slot |
*cmdbuflock[8] | Codec transmitter | Channel/time slot (Ignored in Monitor mode) |
*cmdbuflock[9] | Milliwatt transmitter | Channel/time slot (Ignored in Monitor mode) |
*cmdbuflock[10] | Status line 1 | one byte (See Setup byte interpretation on next page.) |
*cmdbuflock[11] | Status line 2 | one byte (See Setup byte interpretation on next page.) |

* Available for line 1 only.

Resynchronize

cmdbuflock[0] = 2 (Board A) or cmdbuflock[0] = 102 (Board B)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Reset

cmdbuflock[0] = 3 (Board A) or cmdbuflock[0] = 103 (Board B)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Channel Functions

cmdbuflock[0] = 4 (Board A) or cmdbuflock[0] = 104 (Board B)

<table>
<thead>
<tr>
<th>cmdbuflock[1]</th>
<th>mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>If request conflicts with current setup, retain current setup.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>If request conflicts with current setup, override current setup.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cmdbuflock[2]</th>
<th>selection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DS0x receive</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Codec receive</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DS0y transmit</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DS0y receive</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Codec transmit</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Milliwatt transmit</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reset transmit channel</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reset receive channel</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Idle data</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Idle signal</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 24</td>
<td>D4/ESF line 1</td>
<td></td>
</tr>
<tr>
<td>1 - 31</td>
<td>CEPT line 1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8, 4, 2, bits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
C PRIMARY RATE INTERFACE LIBRARY FUNCTIONS – CONTINUED

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:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

- 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

- DSO Bit Inversion
  0 = Off
  1 = On

- CRC Enable (CEPT only)
  0 = CRC Enable off
  1 = CRC Enable on

<table>
<thead>
<tr>
<th>Line</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Off</td>
<td>On</td>
</tr>
</tbody>
</table>

Enable Trace

cmdblock[0] = 9 (Board A only)
cmdblock[1] 8 trace bits, interpreted as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

- 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 version of the PRI library.

char *Pri_version();
ASYNC LIBRARY QUICK REFERENCE

<table>
<thead>
<tr>
<th>ASYNC LIBRARY FILENAME: libasc.a</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INITP1</strong></td>
</tr>
<tr>
<td>int initp1(type, encode)</td>
</tr>
<tr>
<td>1=DTE</td>
</tr>
<tr>
<td>struct ASC_CTRL</td>
</tr>
<tr>
<td>int parity;</td>
</tr>
<tr>
<td>int data;</td>
</tr>
<tr>
<td>int block;</td>
</tr>
<tr>
<td>bitrate</td>
</tr>
<tr>
<td>2 75 8 2400</td>
</tr>
<tr>
<td>3 110 9 4800</td>
</tr>
<tr>
<td>4 150 10 9600</td>
</tr>
<tr>
<td>5 300 11 19200</td>
</tr>
<tr>
<td>6 600</td>
</tr>
<tr>
<td>parity</td>
</tr>
<tr>
<td>1 Odd</td>
</tr>
<tr>
<td>2 Even</td>
</tr>
<tr>
<td>stop</td>
</tr>
<tr>
<td>1 1.5 Stop bits</td>
</tr>
<tr>
<td>2 2 Stop bits</td>
</tr>
<tr>
<td>data</td>
</tr>
<tr>
<td>6 6 Data bits</td>
</tr>
<tr>
<td>7 7 Data bits</td>
</tr>
<tr>
<td>8 8 Data bits</td>
</tr>
<tr>
<td>duplex</td>
</tr>
<tr>
<td>1 Half duplex</td>
</tr>
<tr>
<td>block</td>
</tr>
<tr>
<td>1 Character mode</td>
</tr>
<tr>
<td>eob (End of block character): 0–0xFF</td>
</tr>
</tbody>
</table>

| 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 |
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;
        char sync1;
        char sync2;
        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
C ANALYSIS LIBRARY FUNCTIONS – CONTINUED

BSC If Protocol=8 (BSC), the following parameters must be initialized:

- `par->pbsync.table`: ASCII, EBCDIC
- `par->pbsync.bcc`: CRC16, LRC
- `par->pbsync.sync1` Range: 0x0 to 0xff
- `par->pbsync.sync2` Range: 0x0 to 0xff

AND if `par->pbsync.table` is initialized to ASCII, the following parameter must also be initialized:

- `par->pbsync.parity`: None, Odd, 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 <cham.h>. It is defined as follows:

```c
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 bufsize; /*event.bufsize Data buffer length*/
    unsigned char *pdata; /*event.pdata Data buffer address that points to the frame*/
    long seconds; /*event.seconds Number 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(event)
{
    event *event;
    Returns 0 Successful
    -1 No new events
    -2 Data overwritten (buffer wrapped)
}

reset_anal Resets the acquisition processor.

int reset_anal()
### Multi-Link LAPD Library Quick Reference

**find_link**  
Returns the number of the lowest link matching the SAPI/TEI/TGI value specified.

```c
int find_link(sapi, tei, tgi)
```

Returns:
- 0 - 63: Matching link number
- -1: No match found

**get_freelink**  
Gets the number of first disabled link.

```c
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.

```c
int get_fwaiting (lnkn)
```

Returns:
- 0 - 7: No. of I-frames
- -1: initp1 not performed

**get_link**  
Gets the number of the link currently under user control.

```c
int get_link()
```

Returns:
- 0 - 63: Current link number
- -1: initp1 not performed

**get_inksapi**  
Gets the SAPI value for lnkn.

```c
int get_inksapi (lnkn)
```

Returns:
- 0 - 63: SAPI value
- > 63: Disabling SAPI value

**get_inktei**  
Gets the TEI value for link lnkn.

```c
int get_inktei (lnkn)
```

Returns:
- 0 - 127: TEI value
- > 127: Disabling TEI value

**get_inktgi**  
Gets the TGI value for link lnkn.

```c
int get_inktgi (lnkn)
```

Returns:
- 0 - 14: TGI value
- 15-255: Disabling TGI value

**get_meswaiting**  
Gets no. of messages waiting to be received from the FEP.

```c
int get_meswaiting()
```

Returns:
- 0 - 32: No. of msgs.

**get_rlink**  
Gets the number of the link which sent the last received message.

```c
int get_rlink()
```

Returns:
- 0 - 63: Current link no.
- -1: No messages rec'd
- -2: initp1 not performed

**get_rtei**  
Dummy function to maintain compatibility with single link LAPD programs that are being upgraded to Multi-Link LAPD.

```c
int get_rtei (val)
```

**get_rsapi**  
Dummy function to maintain compatibility with the existing single link LAPD programs that are being upgraded to Multi-Link LAPD.

```c
int get_rsapi (val)
```

**get_rxstat**  
Gets the low order byte of the frame status byte frstat for the last received message.

```c
char get_rxstat()
```

Returns:
- 0-0xC3: frstat value
- 0xFF: No messages rec'd
- 0xFE: initp1 not performed

**get_sapi**  
Gets the SAPI value of the link currently under user control.

```c
int get_sapi()
```

Returns: 0 - 255 SAPI value

**get_sconfig**  
Returns a copy of the current control configuration byte.

```c
int get_sconfig()
```

**get_sim**  
Returns a copy of the network/subscriber selection.

```c
int get_sim()
```

Returns: 0 Network
- 1 Subscriber

**get_tei**  
Gets the TEI of the link currently under user control.

```c
int get_tei()
```

Returns: 0 - 255 TEI value

**get_tgi**  
Gets the TGI of the link currently under user control.

```c
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.

```c
int get_window (lnkn)
```

Returns: 0 - 7 No. of I-frames
MULTI–LINK LAPD LIBRARY (CONTINUED)

initpl

initpl 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 initpl (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;

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;

Returns:
0 – 63
0 Successful
–1 Parameter error
–2 initpl 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;

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 initpl 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;

> 127 disables link

Returns:
0 Successful
–1 Parameter error
–2 initpl not performed
–3 Timeout

89
### Multi-Link Lapd Library Quick Reference (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_tgli</td>
<td>Sets the TGI value for the link under user control.</td>
<td><code>int set_tgli(int v)</code>; <code>char v</code>; 0 to 14 TGI value, 15 to 255 Diables use of TGI</td>
</tr>
<tr>
<td>Returns:</td>
<td>0 Successful; -1 Parameter error; -2 initp1 not performed; -3 Timeout</td>
<td></td>
</tr>
<tr>
<td>set_window</td>
<td>Sets the maximum number of outstanding frames on each link.</td>
<td><code>int set_window(int val)</code>; <code>int val</code>; 1 - 7</td>
</tr>
<tr>
<td>Returns:</td>
<td>0 Successful</td>
<td></td>
</tr>
<tr>
<td>setflg</td>
<td>Selects an interframe fill pattern.</td>
<td><code>int setflg(int flag)</code>; <code>int flag</code>; 0 0x7E fill, 1 0xFF fill</td>
</tr>
<tr>
<td>Returns:</td>
<td>0 Successful</td>
<td></td>
</tr>
<tr>
<td>slof</td>
<td>Disconnects the link.</td>
<td><code>int slof()</code></td>
</tr>
<tr>
<td>Returns:</td>
<td>0 Successful</td>
<td></td>
</tr>
<tr>
<td>slon</td>
<td>Attempts to establish a link.</td>
<td><code>int slon()</code></td>
</tr>
<tr>
<td>Returns:</td>
<td>0 Successful</td>
<td></td>
</tr>
<tr>
<td>srch_link</td>
<td>Returns the number of lowest link matching the specified SAP/TEI.</td>
<td><code>int srch_link(int sapi, tei)</code>; <code>int sapi, tei</code></td>
</tr>
<tr>
<td>Returns:</td>
<td>0 - 63 No. of lowest link; -1 No match found</td>
<td></td>
</tr>
<tr>
<td>tracnl</td>
<td>Transmits an XID command frame with no data field.</td>
<td><code>int tracnl()</code></td>
</tr>
<tr>
<td>Returns:</td>
<td>0 Successful</td>
<td></td>
</tr>
<tr>
<td>trxidc</td>
<td>Transmits a message in an XID command frame.</td>
<td><code>int trxidc(char *xloc, int xlen)</code>; <code>char *xloc</code>, <code>int xlen</code></td>
</tr>
<tr>
<td>Returns:</td>
<td>0 Successful</td>
<td></td>
</tr>
<tr>
<td>trxldr</td>
<td>Transmits a message in an XID response frame.</td>
<td><code>int trxldr(char *xloc, int xlen)</code>; <code>char *xloc</code>, <code>int xlen</code></td>
</tr>
<tr>
<td>Returns:</td>
<td>0 Successful</td>
<td></td>
</tr>
<tr>
<td>trxmi</td>
<td>Transmits an XID response frame with no data field.</td>
<td><code>int trxmi()</code></td>
</tr>
<tr>
<td>Returns:</td>
<td>0 Successful</td>
<td></td>
</tr>
</tbody>
</table>

### Example Code Snippets

```c
int set_tgli(v)
{
    char v;
    0 to 14 TGI value
    15 to 255 Diables use of TGI
    Returns: 0 Successful
    -1 Parameter error
    -2 initp1 not performed
    -3 Timeout
}
```
V.120 LIBRARY

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_FWATING  Gets the number of frames waiting to be transmitted on link.  
    int get_fwating (Inkn)  
    char Inkn;  0 - 63  
    Returns:  0 - 7  No. of 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_lll()  
    Returns:  
        0 - 63  Current link number  

GET_LNKLLI Gets the LLI value for link Inkn.  
    int get_lnklll (Inkn)  
    char Inkn;  0 - 63  
    Returns:  0 - 0x1fff  LLI value  
        >0x1fff  Link Inkn 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 readc  
        -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 - 0x3C  frstat value  
        0xFF  No messages readc  
        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 frames on link number Inkn.  
    int get_window (Inkn)  
    char Inkn;  0 - 63  
    Returns:  0 - 7  No. of frames  

INITP1 Starts the simulator (same as start_sim).  
    int initp1 (interface, sta, encode, 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
V.120 LIBRARY (CONTINUED)

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

SET_LLI Sets the LLI value for the link under user control. A value over 0xFFFF disables the link.
  int set_lli(val)
  int val: 0x00 - 0xFFFF hex
  Returns: 0 Successful

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 Ox7E fill
  1 OxFF 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_LINK Returns the number of lowest link matching the specified SAPI/TEI.
  int srch_link(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 I-frame.
  int transmit(xloc, xlen)
  char *xloc;
  int xlen;

TRANS_RESP Transmits a message in a sequenced-I-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;

TRXCN1 Transmits an XID command frame with no data field.
  int trxcn1()

TRXIDC Transmits a message in an XID command frame.
  int trxidc(xloc, xlen)
  char *xloc;
  int xlen;

TRXIDR Transmits 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**

**MULTI-LINK HDLC LIBRARY**

**Filename:** 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)`

Returns: 1 Successful

**init_b** Initializes Port B.

`int init_b(interface, sta, encode, bitrt)`

Returns: 0 Successful

**initp1** Initializes Ports A and B.

`int initp1(interface, sta, encode, bitrt)`

Returns: 0 Successful

**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

**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

**mlh_set_net** Sets the specified port to act as a network.

`int mlh_set_net(port)`

`int port;`  
`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
mhh_set_t2  Sets the value of the T2 timer
for the specified port.
int mhh_set_t2 (port,val)
port 0 Port A
    1 Port B
val T2 value (1 to 255)
Returns: 0 Successful
         -1 Parameter error

mhh_slof  Disconnects the link on the specified
port.
int mhh_slof (port)
port 0 Port A
    1 Port B

mhh_slon  Attempts to establish a link on the spe-
cified port.
int mhh_slon (port)
port 0 Port A
    1 Port B

mhh_status  Returns the link status of the
specified port.
int mhh_status (port)
port 0 Port A
    1 Port B
Returns: 0 Disconnected
         1 Link conn. requested
         2 Frame reject state
         3 Link disconnect req.
         4 Information xfer state
         5 Local station busy
         6 Remote station busy
         7 Local & remote station
busy

mhh_set_sub  Sets the specified port to act
as a subscriber.
int mhh_set_sub (port)
port 0 Port A
    1 Port B
Returns: 0 Successful

mhh_set_window  Sets the window size for the
specified port.
int mhh_set_window (port,val)
port 0 Port A
    1 Port B
val Window size (1-7)
Returns: 0 Successful
         -1 Parameter error

mhh_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 mhh_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 spe-
cified 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 buff-
er
         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 network.
int set_net
Returns: 0 Successful

set_pat  Specifies a user defined distribution
pattern for transmitting packets using mhh_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
### set_ratio
Selects a distribution pattern for transmitting packets using `mlh_trans()`. It specifies the percentage of packets to be transmitted over Port A.

```c
int set_ratio (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.

```c
int set_sub ()
```

* Returns: 0 - Successful

### set_t1
Sets the T1 timer to an identical value for both ports.

```c
int set_t1 (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.

```c
int set_t2 (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.

```c
int set_window (val)
```

* `val` - Window size (1 to 7)

* Returns: 0 - Successful
  - -1 - Parameter error

### slot
Disconnects the link on both ports.

```c
int slot ()
```

### sion
Attempts to establish a link on both ports by sending a SABM.

```c
int sion ()
```

### status
Returns the link status of the currently selected port.

```c
int status ()
```

* Returns:
  - 0 - Disconnected
  - 1 - Link conn. requested
  - 2 - Frame reject
  - 3 - Link disconn. req.
  - 4 - Information xfer
  - 5 - Local station busy
  - 6 - Remote station busy
  - 7 - Local & remote stations busy

### transmit
Transmits a packet over the specified port.

```c
int transmit (port,xloc,xlen)
```

* `port` - Port A or Port B
  - 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

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)

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)

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)

getFileChoice This function displays a list of files. The function reads the directory specified by the path for each occurrence of a file with the specified extension. For each occurrence, the filename is loaded into the list selector.

getFileChoice (boxName, pPath, ext, bTitle, errMsg, insFlag, inserts, fnum, conf)

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

dsp_req Displays text within a window. The structure type DSPREQ is used to specify the information to be displayed.

derase_field Request to erase a specific field from a window. This will erase both the description and the associated value.

deraseb_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 ERASEREQ.

eraseEOS This function erases the screen from line 3 downward. It is useful in conjunction with pull down menu logic.

eraseEOS()

Returns: None
APPLICATION PROGRAMMER’S INTERFACE C LIBRARY

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;
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,...} 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:

WINDOW_REQ
FIELD
FIELD_DEF
FIELD_SEQ
MB_MESS structure:

```c
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_FLUSH flush the reception buffer

info User defined field when type = CT_ERR or type = CT_CMD

len Length of data if type = CT_DATA

pdata Pointer to the data received or sent.

The library provides the following functions:

**com_chkmb** Checks for received messages.
```c
#include "com.h"
int com_chkmb(pmess)
MB_MESS *pmess;
returns 0 if message received, -1 if no message received.
```

**com_crctlmb** Called by the control task to create the communication channels for the control task.
```c
#include "com.h"
int com_crctlmb(np)
int np;
returns 0 if successful, -1 if insufficient system resources.
```

**com_crpmb** Called by the protocol tasks to create the communication channels for the protocol tasks.
```c
#include "com.h"
int com_crpmb(port,np)
int port;
returns 0 if successful, -1 if insufficient system resources.
```

**com_dlctimb** Closes communication channels. It must be called by the control task prior to terminating.
```c
#include "com.h"
void com_dlctimb();
returns None.
```

**com_error** Reports an error to the control task and is called by a protocol task.
```c
#include "com.h"
void com_error(port,status)
int port;
int status;
returns None.
```

**com_exit** Sends an EXIT message to a protocol task. It is called by the control task.
```c
#include "com.h"
void com_exit(port,lid)
int port;
long lid;
returns None.
```

98
### TASK COMMUNICATION C LIBRARY (CONTINUED)

**com_flush**
Flushes the reception channel and then sends a flush message to the protocol task.

```c
#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.

```c
#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**
 Releasing memory allocated for data messages.

```c
#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.

```c
#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.

```c
#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

returns 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.

```c
#include "com.h"
#include "mtosux.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.

```c
#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: \(\pm 25\) V
- Decision Threshold: \(\pm 3\) V

Line Transmitter:
- Impedance: \(<100\) ohms
- Output Voltage: \(\pm 12\) 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**

<table>
<thead>
<tr>
<th>DB25 Pin No.</th>
<th>CCITT Circuit No.</th>
<th>EIA</th>
<th>Ground</th>
<th>Incoming</th>
<th>Out-going</th>
<th>Processed by Chameleon</th>
<th>RS232 NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>AA</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Frame Ground</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
<td>BA</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Transmitted Data</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>BB</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>Received Data</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>CA</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Request to Send</td>
</tr>
<tr>
<td>5</td>
<td>106</td>
<td>CB</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>6</td>
<td>107</td>
<td>CC</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>Data set Ready</td>
</tr>
<tr>
<td>7</td>
<td>108</td>
<td>AB</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>8</td>
<td>109</td>
<td>CF</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Data Carrier Detect</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+ dc Test Voltage</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- dc Test Voltage</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unassigned</td>
</tr>
<tr>
<td>12</td>
<td>122</td>
<td>SCF</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>2nd Data Carrier Detect</td>
</tr>
<tr>
<td>13</td>
<td>121</td>
<td>SCB</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>2nd Clear to Send</td>
</tr>
<tr>
<td>14</td>
<td>118</td>
<td>SBA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd Transmitted Data</td>
</tr>
<tr>
<td>15</td>
<td>114</td>
<td>DB</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Transmitted Clock</td>
</tr>
<tr>
<td>16</td>
<td>119</td>
<td>SBB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd Received Data</td>
</tr>
<tr>
<td>17</td>
<td>115</td>
<td>DD</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>Receiver Clock</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Receiver Dibit Clock</td>
</tr>
<tr>
<td>19</td>
<td>120</td>
<td>SCA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd Request to Send</td>
</tr>
<tr>
<td>20</td>
<td>108.2</td>
<td>CD</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>21</td>
<td>110</td>
<td>CG</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Signal Quality Detect</td>
</tr>
<tr>
<td>22</td>
<td>125</td>
<td>CE</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Ring Indicator</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data Rate Select</td>
</tr>
<tr>
<td>24</td>
<td>113</td>
<td>DA</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Ext. Transmitter Clock</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Busy</td>
</tr>
</tbody>
</table>

*V.24 PIN ASSIGNMENTS*

Monitoring Mode

<table>
<thead>
<tr>
<th>DB25 Pin No.</th>
<th>CCITT Circuit No.</th>
<th>EIA</th>
<th>Ground</th>
<th>Incoming</th>
<th>Out-going</th>
<th>Processed by Chameleon</th>
<th>RS232 NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>AA</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Frame Ground</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
<td>BA</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Transmitted Data</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>BB</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>Received Data</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>CA</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Request to Send</td>
</tr>
<tr>
<td>5</td>
<td>106</td>
<td>CB</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>6</td>
<td>107</td>
<td>CC</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>Data set Ready</td>
</tr>
<tr>
<td>7</td>
<td>108</td>
<td>AB</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>8</td>
<td>109</td>
<td>CF</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Data Carrier Detect</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+ dc Test Voltage</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- dc Test Voltage</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unassigned</td>
</tr>
<tr>
<td>12</td>
<td>122</td>
<td>SCF</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>2nd Data Carrier Detect</td>
</tr>
<tr>
<td>13</td>
<td>121</td>
<td>SCB</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>2nd Clear to Send</td>
</tr>
<tr>
<td>14</td>
<td>118</td>
<td>SBA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd Transmitted Data</td>
</tr>
<tr>
<td>15</td>
<td>114</td>
<td>DB</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Transmitted Clock</td>
</tr>
<tr>
<td>16</td>
<td>119</td>
<td>SBB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd Received Data</td>
</tr>
<tr>
<td>17</td>
<td>115</td>
<td>DD</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>Receiver Clock</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Receiver Dibit Clock</td>
</tr>
<tr>
<td>19</td>
<td>120</td>
<td>SCA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd Request to Send</td>
</tr>
<tr>
<td>20</td>
<td>108.2</td>
<td>CD</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>21</td>
<td>110</td>
<td>CG</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Signal Quality Detect</td>
</tr>
<tr>
<td>22</td>
<td>125</td>
<td>CE</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Ring Indicator</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data Rate Select</td>
</tr>
<tr>
<td>24</td>
<td>113</td>
<td>DA</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Ext. Transmitter Clock</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Busy</td>
</tr>
</tbody>
</table>

*V.24 PIN ASSIGNMENTS*

Monitoring Mode

<table>
<thead>
<tr>
<th>DB25 Pin No.</th>
<th>CCITT Circuit No.</th>
<th>EIA</th>
<th>Ground</th>
<th>Incoming</th>
<th>Out-going</th>
<th>Processed by Chameleon</th>
<th>RS232 NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>AA</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Frame Ground</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
<td>BA</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Transmitted Data</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>BB</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>Received Data</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>CA</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Request to Send</td>
</tr>
<tr>
<td>5</td>
<td>106</td>
<td>CB</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>6</td>
<td>107</td>
<td>CC</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>Data set Ready</td>
</tr>
<tr>
<td>7</td>
<td>108</td>
<td>AB</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>8</td>
<td>109</td>
<td>CF</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Data Carrier Detect</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+ dc Test Voltage</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- dc Test Voltage</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unassigned</td>
</tr>
<tr>
<td>12</td>
<td>122</td>
<td>SCF</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>2nd Data Carrier Detect</td>
</tr>
<tr>
<td>13</td>
<td>121</td>
<td>SCB</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>2nd Clear to Send</td>
</tr>
<tr>
<td>14</td>
<td>118</td>
<td>SBA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd Transmitted Data</td>
</tr>
<tr>
<td>15</td>
<td>114</td>
<td>DB</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Transmitted Clock</td>
</tr>
<tr>
<td>16</td>
<td>119</td>
<td>SBB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd Received Data</td>
</tr>
<tr>
<td>17</td>
<td>115</td>
<td>DD</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>Receiver Clock</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Receiver Dibit Clock</td>
</tr>
<tr>
<td>19</td>
<td>120</td>
<td>SCA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2nd Request to Send</td>
</tr>
<tr>
<td>20</td>
<td>108.2</td>
<td>CD</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>21</td>
<td>110</td>
<td>CG</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Signal Quality Detect</td>
</tr>
<tr>
<td>22</td>
<td>125</td>
<td>CE</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Ring Indicator</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data Rate Select</td>
</tr>
<tr>
<td>24</td>
<td>113</td>
<td>DA</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>Ext. Transmitter Clock</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Busy</td>
</tr>
</tbody>
</table>
### V.24 PIN ASSIGNMENTS

**Simulation Mode**

<table>
<thead>
<tr>
<th>DB25 Pin No.</th>
<th>CCITT Circuit No.</th>
<th>EIA</th>
<th>Ground</th>
<th>To DCE</th>
<th>From DCE</th>
<th>Processed by Chameleon</th>
<th>RS232 NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>AA</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>Frame Ground</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
<td>BA</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>Transmitted Data</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>BB</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>Received Data</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>CA</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>Request to Send</td>
</tr>
<tr>
<td>5</td>
<td>106</td>
<td>CB</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>Clear to Send</td>
</tr>
<tr>
<td>6</td>
<td>107</td>
<td>CC</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>Data set Ready</td>
</tr>
<tr>
<td>7</td>
<td>108</td>
<td>AB</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>Signal Ground</td>
</tr>
<tr>
<td>8</td>
<td>109</td>
<td>CF</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>Data Carrier Detect</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+ dc Test Voltage</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- dc Test Voltage</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unassigned</td>
</tr>
<tr>
<td>12</td>
<td>122</td>
<td>SCF</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>2nd Data Carrier Detect</td>
</tr>
<tr>
<td>13</td>
<td>121</td>
<td>SCB</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>2nd Clear to Send</td>
</tr>
<tr>
<td>14</td>
<td>118</td>
<td>SBA</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>2nd Transmitted Data</td>
</tr>
<tr>
<td>15</td>
<td>114</td>
<td>DB</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>Transmitted Clock</td>
</tr>
<tr>
<td>16</td>
<td>119</td>
<td>SBB</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>2nd Received Data</td>
</tr>
<tr>
<td>17</td>
<td>115</td>
<td>DD</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>Receiver Clock</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Receiver Dibit Clock</td>
</tr>
<tr>
<td>19</td>
<td>120</td>
<td>SCA</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>2nd Request to Send</td>
</tr>
<tr>
<td>20</td>
<td>108,2</td>
<td>CD</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>21</td>
<td>110</td>
<td>CG</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>Signal Quality Detect</td>
</tr>
<tr>
<td>22</td>
<td>125</td>
<td>CE</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>Ring Indicator</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data Rate Select</td>
</tr>
<tr>
<td>24</td>
<td>113</td>
<td>DA</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>Ext. Transmitter Clock</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Busy</td>
</tr>
</tbody>
</table>
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.

Chameleon connector.
(Female end shown.)

Host Connector.
(Male end shown.)
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

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>CCITT Circuit No.</th>
<th>Name</th>
<th>From DCE</th>
<th>From DTE</th>
<th>Type</th>
<th>RS232 Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>FG</td>
<td></td>
<td></td>
<td></td>
<td>Frame Ground</td>
</tr>
<tr>
<td>B</td>
<td>102</td>
<td>SG</td>
<td></td>
<td></td>
<td></td>
<td>Signal Ground</td>
</tr>
<tr>
<td>C</td>
<td>105</td>
<td>RTS</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Request to Send</td>
</tr>
<tr>
<td>D</td>
<td>106</td>
<td>CTS</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>E</td>
<td>107</td>
<td>DSR</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Data Set Ready</td>
</tr>
<tr>
<td>F</td>
<td>109</td>
<td>DCD</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Data Carrier Detect</td>
</tr>
<tr>
<td>H</td>
<td>108</td>
<td>DTR</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>J</td>
<td>125</td>
<td>RI</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Ring Indicator</td>
</tr>
<tr>
<td>P</td>
<td>103</td>
<td>TD (A)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>R</td>
<td>104</td>
<td>RD (A)</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Receive Data</td>
</tr>
<tr>
<td>S</td>
<td>103</td>
<td>TD (B)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>T</td>
<td>104</td>
<td>RD (B)</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Receive Data</td>
</tr>
<tr>
<td>U</td>
<td>113</td>
<td>SCTE (A)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Transmitter Signal Timing</td>
</tr>
<tr>
<td>V</td>
<td>115</td>
<td>SCR (A)</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Receiver Signal Timing</td>
</tr>
<tr>
<td>W</td>
<td>113</td>
<td>SCTE (B)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Transmitter Signal Timing</td>
</tr>
<tr>
<td>X</td>
<td>115</td>
<td>SCR (B)</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Receiver Signal Timing</td>
</tr>
<tr>
<td>Y</td>
<td>114</td>
<td>SCT (A)</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Transmitter Signal Timing</td>
</tr>
<tr>
<td>AA/a</td>
<td>114</td>
<td>SCT (B)</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Transmitter Signal Timing</td>
</tr>
</tbody>
</table>
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:

![Pinout Diagram]

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

<table>
<thead>
<tr>
<th>DB9 Pin Number</th>
<th>ISO Circuit</th>
<th>CCITT Circuit Mnemonic and Name</th>
<th>Circuit Direction</th>
<th>Circuit Type</th>
<th>Implemented by Chameleon</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>102</td>
<td>SG Signal ground</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>37</td>
<td>102a</td>
<td>SC Send Common</td>
<td>To DCE</td>
<td>Common</td>
<td>X</td>
</tr>
<tr>
<td>20</td>
<td>102b</td>
<td>RC Receive Common</td>
<td>From DCE</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>28</td>
<td>135</td>
<td>IS Terminal in Service</td>
<td>To DCE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>125</td>
<td>IC Incoming Call</td>
<td>From DCE</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>12/30</td>
<td>108</td>
<td>TR Terminal Ready</td>
<td>To DCE</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>11/29</td>
<td>107</td>
<td>DM Data Mode</td>
<td>From DCE</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>4/22</td>
<td>103</td>
<td>SD Send Data</td>
<td>To DCE</td>
<td>Data</td>
<td>P X</td>
</tr>
<tr>
<td>6/24</td>
<td>104</td>
<td>RD Receive Data</td>
<td>From DCE</td>
<td></td>
<td>R X</td>
</tr>
<tr>
<td>17/35</td>
<td>113</td>
<td>TT Terminal Timing</td>
<td>To DCE</td>
<td></td>
<td>I X</td>
</tr>
<tr>
<td>5/23</td>
<td>114</td>
<td>ST Send Timing</td>
<td>From DCE</td>
<td>Timing</td>
<td>M X</td>
</tr>
<tr>
<td>8/26</td>
<td>115</td>
<td>RT Receive Timing</td>
<td>From DCE</td>
<td></td>
<td>A X</td>
</tr>
<tr>
<td>7/25</td>
<td>105</td>
<td>RS Request to Send</td>
<td>To DCE</td>
<td></td>
<td>R X</td>
</tr>
<tr>
<td>9/27</td>
<td>106</td>
<td>CS Clear to Send</td>
<td>From DCE</td>
<td></td>
<td>Y X</td>
</tr>
<tr>
<td>13/31</td>
<td>109</td>
<td>RR Receiver Ready</td>
<td>From DCE</td>
<td></td>
<td>C X</td>
</tr>
<tr>
<td>33</td>
<td>110</td>
<td>SQ Signal Quality</td>
<td>From DCE</td>
<td>Control</td>
<td>H X</td>
</tr>
<tr>
<td>34</td>
<td>136</td>
<td>NS New Signal</td>
<td>To DCE</td>
<td></td>
<td>A N</td>
</tr>
<tr>
<td>16</td>
<td>111/126</td>
<td>SF Select Frequency</td>
<td>To DCE</td>
<td></td>
<td>N E</td>
</tr>
<tr>
<td>16</td>
<td>111/126</td>
<td>SR Signaling Rate Selector</td>
<td>To DCE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>112</td>
<td>SI Signaling Rate Indicator</td>
<td>From DCE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>141</td>
<td>LL Local Loopback</td>
<td>To DCE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>140</td>
<td>RL Remote Loopback</td>
<td>To DCE</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>142</td>
<td>TM Test Mode</td>
<td>From DCE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>116</td>
<td>SS Select Standby</td>
<td>To DCE</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>117</td>
<td>SB Standby Indicator</td>
<td>From DCE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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-subminiature 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:

```
1 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20
```

Electrical Characteristics

RS422 is a Balanced Voltage Digital Signal with electrical characteristics which conform to the CCITT's V.11/X.27.

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver</td>
<td></td>
</tr>
<tr>
<td>Output resistance:</td>
<td>200 ohms differential</td>
</tr>
<tr>
<td>Lead to ground:</td>
<td>175 ohms</td>
</tr>
<tr>
<td>Output current:</td>
<td>150 mA maximum</td>
</tr>
<tr>
<td>Output voltage:</td>
<td>±3 volts</td>
</tr>
<tr>
<td>Receiver</td>
<td></td>
</tr>
<tr>
<td>Input resistance:</td>
<td>200 ohms differential</td>
</tr>
<tr>
<td>Lead to ground:</td>
<td>175 ohms</td>
</tr>
<tr>
<td>Input sensitivity:</td>
<td>±200 mvolts</td>
</tr>
</tbody>
</table>
## RS422/V.11/V.36 Connector Pinout

<table>
<thead>
<tr>
<th>DB37 Pin Number</th>
<th>ISO Circuit</th>
<th>CCITT Circuit Mnemonic and Name</th>
<th>Circuit Direction</th>
<th>Circuit Type</th>
<th>Implemented by Chameleon</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>102</td>
<td>SG Signal ground</td>
<td>-</td>
<td>Common</td>
<td>X</td>
</tr>
<tr>
<td>37</td>
<td>102a</td>
<td>SC Send Common</td>
<td>To DCE</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>20</td>
<td>102b</td>
<td>RC Receive Common</td>
<td>From DCE</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>28</td>
<td>135</td>
<td>IS Terminal in Service</td>
<td>To DCE</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>125</td>
<td>IC Incoming Call</td>
<td>From DCE</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>12 / 30</td>
<td>108</td>
<td>TR Terminal Ready</td>
<td>To DCE</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>11 / 29</td>
<td>107</td>
<td>DM Data Mode</td>
<td>From DCE</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4 / 22</td>
<td>103</td>
<td>SD Send Data</td>
<td>To DCE</td>
<td>P</td>
<td>X</td>
</tr>
<tr>
<td>6 / 24</td>
<td>104</td>
<td>RD Receive Data</td>
<td>From DCE</td>
<td>P</td>
<td>X</td>
</tr>
<tr>
<td>17 / 35</td>
<td>113</td>
<td>TT Terminal Timing</td>
<td>To DCE</td>
<td>I</td>
<td>X</td>
</tr>
<tr>
<td>5 / 23</td>
<td>114</td>
<td>ST Send Timing</td>
<td>From DCE</td>
<td>M</td>
<td>X</td>
</tr>
<tr>
<td>8 / 26</td>
<td>115</td>
<td>RT Receive Timing</td>
<td>From DCE</td>
<td>A</td>
<td>X</td>
</tr>
<tr>
<td>7 / 25</td>
<td>105</td>
<td>RS Request to Send</td>
<td>To DCE</td>
<td>R</td>
<td>X</td>
</tr>
<tr>
<td>9 / 27</td>
<td>106</td>
<td>CS Clear to Send</td>
<td>From DCE</td>
<td>Y</td>
<td>X</td>
</tr>
<tr>
<td>13 / 31</td>
<td>109</td>
<td>RR Receiver Ready</td>
<td>From DCE</td>
<td>C</td>
<td>X</td>
</tr>
<tr>
<td>33</td>
<td>110</td>
<td>SQ Signal Quality</td>
<td>From DCE</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>136</td>
<td>NS New Signal</td>
<td>To DCE</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>111 / 126</td>
<td>SF Select Frequency</td>
<td>To DCE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>111 / 126</td>
<td>SR Signaling Rate Selector</td>
<td>To DCE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>112</td>
<td>SI Signaling Rate Indicator</td>
<td>From DCE</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>141</td>
<td>LL Local Loopback</td>
<td>To DCE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>140</td>
<td>RL Remote Loopback</td>
<td>To DCE</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>142</td>
<td>TM Test Mode</td>
<td>From DCE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>116</td>
<td>SS Select Standby</td>
<td>To DCE</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>117</td>
<td>SB Standby Indicator</td>
<td>From DCE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Pin Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/STROBE (Active Low)</td>
</tr>
<tr>
<td>2</td>
<td>Data 0</td>
</tr>
<tr>
<td>3</td>
<td>Data 1</td>
</tr>
<tr>
<td>4</td>
<td>Data 2</td>
</tr>
<tr>
<td>5</td>
<td>Data 3</td>
</tr>
<tr>
<td>6</td>
<td>Data 4</td>
</tr>
<tr>
<td>7</td>
<td>Data 5</td>
</tr>
<tr>
<td>8</td>
<td>Data 6</td>
</tr>
<tr>
<td>9</td>
<td>Data 7</td>
</tr>
<tr>
<td>10</td>
<td>/ACK (Active Low)</td>
</tr>
<tr>
<td>11</td>
<td>Busy</td>
</tr>
<tr>
<td>12</td>
<td>No Connection</td>
</tr>
<tr>
<td>13</td>
<td>No Connection</td>
</tr>
<tr>
<td>14</td>
<td>No Connection</td>
</tr>
<tr>
<td>15</td>
<td>No Connection</td>
</tr>
<tr>
<td>16</td>
<td>No Connection</td>
</tr>
<tr>
<td>17</td>
<td>No Connection</td>
</tr>
<tr>
<td>18</td>
<td>Ground</td>
</tr>
<tr>
<td>19</td>
<td>Ground</td>
</tr>
<tr>
<td>20</td>
<td>Ground</td>
</tr>
<tr>
<td>21</td>
<td>Ground</td>
</tr>
<tr>
<td>22</td>
<td>Ground</td>
</tr>
<tr>
<td>23</td>
<td>Ground</td>
</tr>
<tr>
<td>24</td>
<td>No Connection</td>
</tr>
<tr>
<td>25</td>
<td>Ground</td>
</tr>
</tbody>
</table>
The Chameleon serial printer connector is a 25 pin D-subminiature socket (female) (DB25S). The pinout is shown as the connector is viewed from the rear of the machine:

![Diagram of Chameleon serial printer connector]

All signals are standard RS-232 voltage levels. The connector is physically and electrically a DCE type connector.

<table>
<thead>
<tr>
<th>DB25 Pin No.</th>
<th>CCITT Circuit No.</th>
<th>EIA</th>
<th>Source</th>
<th>Signal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>AA</td>
<td>Chassis</td>
<td>Chassis Ground</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
<td>BA</td>
<td>Printer</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>BB</td>
<td>Chameleon</td>
<td>Receive Data</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>CA</td>
<td>Printer</td>
<td>Request to Send</td>
</tr>
<tr>
<td>5</td>
<td>106</td>
<td>CB</td>
<td>Chameleon</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>6</td>
<td>107</td>
<td>CC</td>
<td>Chameleon</td>
<td>Data Set Ready</td>
</tr>
<tr>
<td>7</td>
<td>102</td>
<td>AB</td>
<td>Signal Gnd.</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>8</td>
<td>109</td>
<td>CF</td>
<td>Chameleon</td>
<td>Carrier Detect</td>
</tr>
<tr>
<td>15</td>
<td>114</td>
<td>DB</td>
<td>Chameleon</td>
<td>Transmit Clock</td>
</tr>
<tr>
<td>17</td>
<td>115</td>
<td>DD</td>
<td>Chameleon</td>
<td>Receive Clock</td>
</tr>
<tr>
<td>20</td>
<td>108</td>
<td>CD</td>
<td>Printer</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>DA</td>
<td>Printer</td>
<td>External Clock</td>
</tr>
</tbody>
</table>

Note: The table lists the pin numbers (DB25 Pin No.), the corresponding circuit numbers (CCITT Circuit No.), the EIA designation (EIA), the source (Source), and the signal names (Signal Name).
The Chameleon Remote I/O connector is a 25 pin D-subminiature 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.

<table>
<thead>
<tr>
<th>DB25 Pin No.</th>
<th>CCITT Circuit No.</th>
<th>EIA</th>
<th>Source</th>
<th>Signal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>AA</td>
<td>Chassis</td>
<td>Chassis Ground</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
<td>BA</td>
<td>Printer</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>BB</td>
<td>Chameleon</td>
<td>Receive Data</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>CA</td>
<td>Printer</td>
<td>Request to Send</td>
</tr>
<tr>
<td>5</td>
<td>106</td>
<td>CB</td>
<td>Chameleon</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>6</td>
<td>107</td>
<td>CC</td>
<td>Chameleon</td>
<td>Data Set Ready</td>
</tr>
<tr>
<td>7</td>
<td>102</td>
<td>AB</td>
<td>Signal Gnd.</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>8</td>
<td>109</td>
<td>CF</td>
<td>Chameleon</td>
<td>Carrier Detect</td>
</tr>
<tr>
<td>15</td>
<td>114</td>
<td>DB</td>
<td>Chameleon</td>
<td>Transmit Clock</td>
</tr>
<tr>
<td>17</td>
<td>115</td>
<td>DD</td>
<td>Chameleon</td>
<td>Receive Clock</td>
</tr>
<tr>
<td>20</td>
<td>108</td>
<td>CD</td>
<td>Printer</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>24</td>
<td>–</td>
<td>DA</td>
<td>Printer</td>
<td>External Clock</td>
</tr>
</tbody>
</table>
AUX 1 AND AUX 2 PORTS
CONNECTOR PINOUTS

The Chameleon Aux 1 and Aux 2 serial port connectors are 25 pin D-subminiature sockets (female) (DB25S). The pinout for both is shown as the connector is viewed from the rear of the machine:

![Port Connector Diagram]

All signals are standard RS232 voltage levels. The connector is physically and electrically a DCE type connector.

<table>
<thead>
<tr>
<th>DB25 Pin No.</th>
<th>CCITT Circuit No.</th>
<th>EIA</th>
<th>Source</th>
<th>Signal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>AA</td>
<td>N/C</td>
<td>Chassis Ground</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
<td>BA</td>
<td>Terminal TXD</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>BB</td>
<td>Chameleon RXD</td>
<td>Receive Data</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>CA</td>
<td>Terminal RTS</td>
<td>Request to Send</td>
</tr>
<tr>
<td>5</td>
<td>106</td>
<td>CB</td>
<td>Chameleon CTS</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>6</td>
<td>107</td>
<td>CC</td>
<td>Chameleon DSR</td>
<td>Data Set Ready</td>
</tr>
<tr>
<td>7</td>
<td>102</td>
<td>AB</td>
<td>Signal Gnd. GND</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>15</td>
<td>114</td>
<td>DB</td>
<td>Chameleon TXC</td>
<td>Transmit Clock</td>
</tr>
<tr>
<td>17</td>
<td>115</td>
<td>DD</td>
<td>Chameleon RXC</td>
<td>Receive Clock</td>
</tr>
<tr>
<td>20</td>
<td>108</td>
<td>CD</td>
<td>Terminal DTR</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>22</td>
<td>125</td>
<td>CE</td>
<td>Terminal RI</td>
<td>Ring Indicator</td>
</tr>
<tr>
<td>24</td>
<td>-</td>
<td>DA</td>
<td>Terminal CK</td>
<td>External Clock</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ground</td>
</tr>
<tr>
<td>2</td>
<td>Ground</td>
</tr>
<tr>
<td>3</td>
<td>Red</td>
</tr>
<tr>
<td>4</td>
<td>Green</td>
</tr>
<tr>
<td>5</td>
<td>Blue</td>
</tr>
<tr>
<td>6</td>
<td>Intensity</td>
</tr>
<tr>
<td>7</td>
<td>Monochrome</td>
</tr>
<tr>
<td>8</td>
<td>Horizontal Sync.</td>
</tr>
<tr>
<td>9</td>
<td>Vertical Sync.</td>
</tr>
</tbody>
</table>

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.
The Chameleon SCSI interface signals are as shown below. All signals are low true.

All odd pins are ground.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GND</td>
<td>2 Data Bit 0 (DB0).</td>
</tr>
<tr>
<td>3</td>
<td>4 Data Bit 1 (DB1).</td>
</tr>
<tr>
<td>5</td>
<td>6 Data Bit 2 (DB2).</td>
</tr>
<tr>
<td>7</td>
<td>8 Data Bit 3 (DB3).</td>
</tr>
<tr>
<td>9</td>
<td>10 Data Bit 4 (DB4).</td>
</tr>
<tr>
<td>11</td>
<td>12 Data Bit 5 (DB5).</td>
</tr>
<tr>
<td>13</td>
<td>14 Data Bit 6 (DB6).</td>
</tr>
<tr>
<td>15</td>
<td>16 Data Bit 7 (DB7).</td>
</tr>
<tr>
<td>17</td>
<td>18 Data Parity (DBP).</td>
</tr>
<tr>
<td>19</td>
<td>20 Open.</td>
</tr>
<tr>
<td>21</td>
<td>22 Open.</td>
</tr>
<tr>
<td>23</td>
<td>24 Open.</td>
</tr>
<tr>
<td>25</td>
<td>26 Open.</td>
</tr>
<tr>
<td>27</td>
<td>28 Open.</td>
</tr>
<tr>
<td>29</td>
<td>30 Open.</td>
</tr>
<tr>
<td>31</td>
<td>32 Open.</td>
</tr>
<tr>
<td>33</td>
<td>34 Open.</td>
</tr>
<tr>
<td>35</td>
<td>36 Busy (BSY).</td>
</tr>
<tr>
<td>37</td>
<td>38 Acknowledge (ACK).</td>
</tr>
<tr>
<td>39</td>
<td>40 Reset (RST).</td>
</tr>
<tr>
<td>41</td>
<td>42 Message (MSG).</td>
</tr>
<tr>
<td>43</td>
<td>44 Select (SEL).</td>
</tr>
<tr>
<td>45</td>
<td>46 Control/Data (C/D).</td>
</tr>
<tr>
<td>47</td>
<td>48 Request (REQ).</td>
</tr>
<tr>
<td>49 GND</td>
<td>50 Input/Output (I/O).</td>
</tr>
</tbody>
</table>
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 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

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 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
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:

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shield</td>
</tr>
<tr>
<td>2</td>
<td>T(A) Transmit (A)</td>
</tr>
<tr>
<td>3</td>
<td>C(A) Control (A)</td>
</tr>
<tr>
<td>4</td>
<td>R(A) Receive (A)</td>
</tr>
<tr>
<td>5</td>
<td>I(A) Indication (A)</td>
</tr>
<tr>
<td>6</td>
<td>S(A) Signal Element Timing (A)</td>
</tr>
<tr>
<td>7</td>
<td>B(A) Byte Timing (A)</td>
</tr>
<tr>
<td>8</td>
<td>Ground</td>
</tr>
<tr>
<td>9</td>
<td>T(B) Transmit (B)</td>
</tr>
<tr>
<td>10</td>
<td>C(B) Control (B)</td>
</tr>
<tr>
<td>11</td>
<td>R(B) Receive (B)</td>
</tr>
<tr>
<td>12</td>
<td>I(B) Indication (B)</td>
</tr>
<tr>
<td>13</td>
<td>S(B) Signal Element Timing (B)</td>
</tr>
<tr>
<td>14</td>
<td>B(B) Byte Timing (B)</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>DA15S Pin Number</th>
<th>RS449 Pin Number</th>
<th>Description</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Chassis Ground</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>Send Data B</td>
<td>Input</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Unused</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>Receive Data B</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Unused</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>23</td>
<td>Send Timing B</td>
<td>Output</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>Signal Ground</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>Send Data A</td>
<td>Inputs</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Unused</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>Receive Data A</td>
<td>Output</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Unused</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>Send Timing A</td>
<td>Output</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Unused</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>Receive Timing A</td>
<td>Output</td>
</tr>
</tbody>
</table>