D.S

M.I.T.E. Version 2.3

The Mycroft Intelligent Terminal Emulator

A Data Communcations System for CP/M

Mycroft Labs, Inc.
P.O. Box 6045
Tallahassee, FL 32301

15 Dec 1982

MYCROFT LABS
Box 6045, Tallahassee, Florida, 32301
SOFTWARE LICENSE AGREEMENT

IMPORTANT: All Mycroft Labs programs are sold only on the condition that the purchaser agrees to the following license. READ THIS LICENSE CAREFULLY. If you do not agree to the terms contained in this license, return the packaged diskette UNOPENED to your distributor and your purchase price will be refunded. If you agree to the terms contained in this license, fill out the REGISTRATION information and RETURN by mail.

MYCROFT LABS agrees to grant and the Customer agrees to accept on the following terms and conditions nontransferrable and nonexclusive licenses to use the software program(s) (Licensed Programs) herein delivered with this agreement.

TERM:

This agreement is effective from the date of receipt of the above-referenced program(s) and shall remain in force until terminated by the Customer upon one month's prior written notice, or by Mycroft Labs as provided below.

Any license under this Agreement may be discontinued by the Customer at any time upon one month's prior written notice. Mycroft Labs may discontinue any license or terminate this Agreement if the Customer fails to comply with any of the terms and conditions of this Agreement.

LICENSE:

Each program license granted under this Agreement authorizes the Customer to use the Licensed Program in any machine readable form on any single computer system (referred to as System). A separate license is required for each System on which the Licensed program will be used.

This Agreement and any of the licenses, programs or materials to which it applies may not be assigned, sublicensed or otherwise transferred by the Customer without prior written consent from Mycroft Labs. No right to print or copy, in whole or in part, the Licensed Programs is granted except as hereinafter expressly provided.

PERMISSION TO COPY OR MODIFY LICENSED PROGRAMS:

The Customer shall not copy, in whole or in part, any Licensed Programs which are provided by Mycroft Labs in printed form under this Agreement. Additional copies of printed materials may be acquired from Mycroft Labs.

Any Licensed Programs which are provided by Mycroft Labs in machine readable form may be copied, in whole or in part, in printed or machine readable form in sufficient number for use by the Customer with the designated System, to understand the contents of such machine readable material, to modify the Licensed Program as provided below, for back-up purposes, or for archive purposes, provided, however, that no more than five (5) printed copies will be in existence under any license at any one time without prior written consent from Mycroft Labs. The Customer agrees to maintain appropriate records of the number and location of all such copies of Licensed Programs. The original, and any copies of the Licensed Programs, in whole or in part, which are made by the Customer shall be the property of Mycroft Labs. This does not imply, of course, that Mycroft Labs owns the media on which the Licensed Programs are recorded. The Customer may modify any machine readable form of the Licensed Programs for his own use and merge it into other program material to form an updated work, provided that, upon discontinuance of the license for such Licensed Program, the Licensed Program supplied by Mycroft Labs will be completely removed from the updated work. Any portion of the Licensed Program included in an updated work shall remain subject to all other terms of this agreement, and Mycroft Labs retains the right to ownership of any such included portions of the Licensed Program, although no charge is made for such inclusion.

The Customer agrees to reproduce and include the copyright notice of Mycroft Labs on all copies, in whole or in part, in any form, including partial copies or modifications, of Licensed Programs made hereunder. The Customer further agrees to never remove any copyright notices from any Mycroft Labs software or documentation in either printed or machine readable form.

PROTECTION AND SECURITY:

The Customer agrees not to provide or otherwise make available any Licensed Program including but not limited to program listings, object code and source code, in any form, to any person other than Customer or Mycroft Labs employees, without prior written consent from Mycroft Labs.

DISCONTINUANCE:

Within one month after the date of discontinuance of any license under this Agreement, the Customer will furnish Mycroft Labs a certificate certifying that through his best effort, and to the best of his knowledge, the original and all copies, in whole or in part, in any form, including partial copies or modifications, of the Licensed Programs received from Mycroft Labs have been destroyed, except that, upon prior written authorization from Mycroft Labs, the Customer may retain a copy for archive purposes.

DISCLAIMER OF WARRANTY:

Mycroft Labs makes no warranties with respect to the Licensed Programs. The sole obligation of Mycroft Labs shall be to make available all published modifications or updates made by Mycroft Labs to Licensed Programs which are published within one (1) year from date of purchase, provided Customer has returned the Registration Card delivered with the Licensed Program.

LIMITATION OF LIABILITY:

THE FOREGOING WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MYCROFT LABS BE LIABLE FOR CONSEQUENTIAL DAMAGES EVEN IF MYCROFT LABS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

GENERAL:

If any of the provisions, or portions thereof, of this Agreement are invalid under any applicable statute or rule of law, they are to that extent to be deemed omitted.

This Agreement is adapted from an original Agreement developed by the legal department of Digital Research, Inc., and is used by permission.

Mycroft Labs Software Registration Form

Software _____ Serial # _____

Name _____

Address _____

City, State, Zip _____

Date Purchased _____

Purchased from _____

Table of Contents

Appendices

Introduction

M.I.T.E. stands for the "Mycroft Intelligent Terminal Emulator". It is a state-of-the-art intelligent terminal and file transfer utility for the CP/M Operating System (tm Digital Research). It is menu-oriented and has a number of "user protection" features, which makes it especially simple to learn and use, even by non-technical personnel. It is intended for use in four primary applications:

1 - Accessing online timesharing systems, such as might be found at many University computing centers or on-line data base utilities. Virtually any system that supports ASCII teletypes (or 'dumb' CRT terminals) can be accessed, with full text file transfer capability. Some of the more popular compatible systems are:

DEC	All models, e.g. PDP-11, Vax-11
CDC	6000 series, Cyber 170 series, etc.
DG	Nova, Eclipse, etc.
PRIME	All models
HONEYWELL	Level 62, Level 6, etc.
HARRIS	All models

Most on-line data base utilities are compatible with MITE, including virtually every service available through TELENET and/or TYMNET. Examples include:

The Source
Compuserve
Dow Jones

2 - Accessing on-line CP/M systems (RCPM) and Computerized Bulletin Board Systems (CBBS - (c) Ward Christensen), and many other such message and/or public domain software systems. MITE supports the standard "XMODEM" protocol, as used on most RCPM systems, for error-free transfer of any CP/M files, including raw object code (.COM files).

3 - File exchange with other CP/M microcomputer systems running MITE, CLINK, or various other "smart terminal programs". MITE supports several protocols, for compatibility with the greatest number of such programs. MITE is also compatible with Radio Shack TRS-80 computers that are running MODEM-80 from The Alternate Source, by using the XMODEM protocol. Note that file formats are not exactly compatible between CP/M and TRSDOS, hence a utility program is included with MITE to convert text files from TRSDOS format to CP/M format.

4 - Accessing the Western Union TWX (or indirectly the TELEX) network.

MITE is available pre-installed on a number of popular CP/M systems, including the following:

Xerox 820 (both 8" SS/SD and 5.25" SS/SD)
Kay Comp II (5.25" SS/SD)
Televideo TS802 (5.25" DS/DD)
Zenith Z89 (5.25" SS/SD 10 sector) and Z90 (5.25" DS/DD)
Radio Shack Model II under CP/M (8" SS/SD)
Sanyo 1000 (5.25" SS/DD)
Sanyo 1200 (5.25" SS/DD/DT)
S100 Systems with PMMI MM-103 modem (8" SS/SD)
S100 Systems with Hayes Micromodem 100 (8" SS/SD)
Apple II with Z80 Softcard and Hayes Micromodem II (5.25")
Superbrain (5.25" SS/DD)
Exxon 500 (5.25" DS/DD 16 sector)
Vector 4 (5.25" DS/DD 16 sector)
Osborne I (5.25" SS/SD)
Eagle II (5.25" SS/DD/DT)

An "uninstalled" version is also available which can be customized for use on a wide range of CP/M systems by writing a "Communications Input Output System" (CIOS), similar to the CP/M BIOS and patching it into MITE with DDT or SID. User customizable CIOS modules for a wide range of serial I/O chips (8251, Z80SIO, ACIA, 8250 ACE, etc.) are included.

All communications parameters (including the phone number and any macro strings) specific to a given site can be selected easily via the menu options, then saved off on a "parameter file" for future use. Once this is done, all parameters can be set in a single operation by specifying this file as an argument on the command line (or via an option on the Main menu). All options can be set with a minimum of hassle, many with a single keystroke.

MITE has numerous "user protection" features to help prevent accidental loss of captured data or disk files. Anytime a file is created (e.g. when capturing a file, or saving parameters), the directory is checked to see if there is already a file with that name. If so, you are given the choice of over-writing it, or aborting the current operation so that a different filename can be specified. When exiting to CP/M, if a capture operation is in progress, the data will be automatically flushed to disk. Also, if the carrier is still present at that time, you are informed of this fact and given the option of hanging up.

Running MITE

MITE must be installed on most systems before it may be run. An uninstalled version MITE/U.COM should be present on the disk in order to install and execute MITE. Please install MITE as per the directions given in "Installation Notes".

To run MITE, just type its name in CP/M command mode. This will result in reasonable default values being used for all parameters, or the parameters may be loaded from a file as an

option on the main menu. Optionally a parameter file may be specified as the first argument:

```
A>mite source
```

In this case, all parameters will be read from the file "SOURCE.PAR". If a filetype is specified on the argument, that will override the default filetype of ".PAR". In either case, at this time the Main menu will be displayed, and any necessary changes to the parameters may be made, and optionally saved for future use.

A second parameter may also be specified, which is taken as the first option on the Main menu, if present. This allows the user to start up directly in answer or originate mode. For example:

```
A>mite cyber g
```

Local Commands:

When the 'Command Trigger' character (option K on the OPTION MENU) has been set to a non-zero value (recommended value = ^K), any time that character is typed on the local console, the prompt:

Local Command?

will be displayed. At this time, any system command (see SYSTEM COMMAND PROCESSOR), or any of the following local commands may be entered. Once that single command has been processed, the transparent link will automatically be resumed. Note that only the first four characters of the local commands need be entered, and if the filename is omitted, MITE will ask for it later. These commands are intended to duplicate existing menu functions in a way that the more sophisticated user will find to be quicker and less intrusive. The following local commands are available:

CAPTure ON/OFF

Allows user to turn text capture mode ON or OFF (see the C option on the TEXT FILE DOWNLOAD MENU). The first time capture is enabled (or the first time after a WRITE), MITE will ask for a filename.

WRITe

Allows the user to write the captured data (or the last part of it if flow control is enabled) to disk. See the W option on the TEXT FILE DOWNLOAD MENU.

ECHO ON/OFF

Allows the user to turn the printer echo function ON or OFF (see the P option on the TEXT FILE DOWNLOAD MENU).

HELP Local

List available local commands to console.

HELP System

List available system commands to console.

READ [d:fn.ft]

Read specified file from disk and send it as if it were coming from the local keyboard (see the U option on the TEXT FILE UPLOAD MENU).

PROTocol

Display current binary protocol and allow new one to be selected.

SEND [d:fn.ft]

Send specified file from disk using the currently selected binary protocol (see the S option on the BINARY FILE TRANSFER MENU).

RECV [d:fn.ft]

Receive file transmission onto specified file using the currently selected binary protocol (see the R option on the BINARY FILE TRANSFER MENU).

MACRos

List all non-blank macro strings to console.

Note that as of version 2.2, it is possible to specify a 'Remote Command' Trigger character. When this is specified as a non-zero value, anytime this character is typed by either the originate or answer system user, the prompt 'Remote Command?' will be displayed, at which time either user can enter a 'local' command, which is then performed on the answer system, and the output for which is sent to BOTH systems.

This mechanism allows you to put a MITE system online so that other MITE users can dial into it and request directories, erase files, check space available, change the protocol and/or send and receive files. In fact, all operations can be done from the originating system, hence the answer system can be left totally unattended. Note that the remote command trigger character (if non-zero) is displayed when the header is printed once a call has been answered. Also note that this capability need only be present on the answer system, hence earlier versions of MITE (or other terminal programs) can serve as the originate system.

General

Note that at the head of each menu the following three lines are displayed:

M.I.T.E. vx.y - Copyright (c) 1982, Mycroft Labs Inc.
XXXXXX. Bytes Captured = nnnnn/nnnnn. Capture = XXX.
Site ID = xxx ... xxx

In the above, the "vx.y" is the current version number (x) and release level (y) of M.I.T.E. The second line lists the status of three things as of the time the header was written. The first field will contain the word ONLINE or OFFLINE. This reflects the current state of the carrier. The second field tells how many bytes have been captured, and how many it is possible to capture in your particular system (e.g. Bytes Captured = 1254/40725 would mean you have currently captured 1254 bytes out of a possible 40725). The third field tells whether capture mode is currently ON or OFF. This status line allows the user to get some quick feedback on these three items, as well as reassuring you that carrier has not been lost when you are in the menu system. The third line is the current Site ID.

Note that as of version 2.0 of M.I.T.E. there is a Western Union TWX compatibility mode. By selecting the "TWX Mode" on the option menu, the operation of M.I.T.E. is modified to allow your microcomputer to serve as a terminal on the TWX network. It is also possible to access any TELEX terminal in the world via either of two mechanisms supported by Western Union.

When the TWX mode of operation is selected, the following differences are in effect:

* Anytime a Ctrl-E is received, macro string number 8 will be sent as an "answerback" message.

* On establishing a connection, MITE will send a ctrl-E, and await the returned "answerback", echoing it to your console. Once this has been seen, MITE will automatically ask for the name of the file to be uploaded, which is then sent. Once the file is sent, control will be returned to the normal transparent link.

* On detection of an incoming call, the normal header is suppressed.

It is recommended that you capture the entire session on disk, or to your printer. Be sure the baud rate is set to 110. Since TWX mode is saved as a parameter, it should be sufficient to create a parameter file called TWX.PAR that will set up everything.

The Main Menu

```
M.I.T.E. v2.x - Copyright (c) 1982, Mycroft Labs Inc.
OFFLINE. Bytes Captured =      0/32267. Capture = OFF.
Site ID = Source Telecomputing Corporation

MAIN MENU

     G - Go Start Communications
     H - Hangup Phone
     I - Enter Site ID
     L - Load Parameters from Disk File
     S - Save Parameters on Disk File

           Sub-Menus:

     P - Parameter              O - Option
     U - Text File Upload       D - Text File Download
     B - Binary File Xfer       M - Macro Definition
     C - Command Processor      F - Character Filter

     X - Exit to CP/M

Enter Option: _
```

The G option does one of several different things, depending on the current setting of the "Role" option (ANS or ORG), and whether or not the carrier is present. First off, if the carrier is present, the transparent link mode is immediately resumed. If the carrier is NOT present, then the procedure depends on the current setting of the "role" option. If "role" is set to ANS, then MITE will await an incoming call (and answer it, assuming the hardware supports auto-answer). In this case, if the carrier is lost, MITE will cycle back to waiting for the next call. If "role" is set to ORG, then MITE will dial the specified phone number (if one is specified, and the hardware supports auto-dial) and wait up to 30 seconds for an incoming carrier.

In any case, once the connection has been established, MITE will enter the transparent link mode. Once this mode is entered, any character typed on the console keyboard will be sent to the remote system, and any character received from the remote system will be written to the console display. The only exceptions to this are the "trigger" characters (escape, macro and break) which the user can select to be any ASCII characters. The "escape trigger" is used to return to the Main menu, the "macro trigger" is used (in conjunction with a second character) to invoke one of the ten macro strings, and the "break trigger" is used to send a "break" function to the remote system.

The I option allows you to enter a one-line description of the site for which these parameters are intended. This "site ID" will be printed on the third line of each menu page. When using MITE in the ANSWER role, this site ID should be set to something identifying YOUR site, as it will be sent to the user dialing into your system, along with the standard MITE greeting.

The H option can be used to hang up the phone at any time. With some on-line systems, this may be the only way to terminate a session. Not all such systems support a "BYE" or "OFF" command that causes their carrier to go away. If you try to exit to CP/M with carrier still present, you will be reminded that it is still present, and asked if you wish to hang up at that time. Normally, when the carrier is lost, the phone is automatically hung up and control returns to the Main menu, from which it is possible to exit to CP/M. Note that the proper functioning of this option depends on the actual implementation. Not all hardware allows the software to hangup the phone.

The L option allows you to load parameters from a previously saved parameter file. You will be prompted for a filename, which should be entered in the "d:fn.ft" format (e.g. SOURCE, RATOFF.1, B:CPMNET, etc.). If no filetype is specified, the default filetype .PAR will be used. If no such file is found, you will be notified, and control returns to the Main menu. If that file is found, the following parameters will be loaded:

Byte	Size	Contents
0	1	ASCII character 'M'
1	1	Version number * 10 + release number
2	20	Phone Number
22	2	Baud Rate (high byte first)
24	1	Parity (0=NONE, 1=ODD, 2=EVEN)
25	1	Number of Data Bits (0=SEVEN, 1=EIGHT)
26	1	Number of Stop Bits (0=ONE, 1=TWO)
27	1	Duplex (0=HALF, 1=FULL)
28	1	Escape Trigger char
29	1	Wait-For-Echo option (0=OFF, 1=ON)
30	1	Flow Control option (0=OFF, 1=ON)
31	1	Flow Control Start Character
32	1	Flow Control Stop Character
33	1	End-Of-Line Handshaking option (0=OFF, 1=ON)
34	1	Auto-LF option (0=OFF, 1=ON)
35	1	Caps option (0=off, 1=ON)
36	1	Macro Trigger Character
37	1	Protocol (0=XMODEM, 1=CLINK, 2=HAYES, 3=IBMPC)
38	60	Site ID
98	1	Garbage Character count
99	1	Turnaround Character
100	1	Intercharacter delay, msec
101	1	Role (0=ORG, 1=ANS)
102	1	Break Trigger Character
103	1	TWX mode flag (0=OFF, 1=ON)
104	1	Printer Echo (0=OFF, 1=ON)

105	1	Command Trigger Character
106	1	Remote Command Trigger Character
107	10	Unwanted Characters
117	11	zeros

The ten macro strings then follow, 64 bytes per string. If anything other than a valid parameter file is specified, MITE will inform you of this fact and abort the load command:

"Invalid Parameter File"

If a parameter file created with an old version of MITE is loaded, then you will be warned:

"Warning - old parameter file."

at which point you should check all parameters and save them back off (on the same file, normally) with the new version of MITE.

The S option allows you to save the current parameters to a disk file for future use as a command line argument or as input for the L option. You will be prompted for a filename, which should be entered in the "d:fn.ft" format. If no filetype is specified, the default filetype .PAR will be used. All parameters listed under the L option will be saved.

Sub-Menus

The remaining options allow you to transfer control to any of the sub-menus, from which you would normally return to the Main menu, once you have accomplished the desired operation(s) on that sub-menu.

The P option selects the PARAMETER MENU. From this menu, you can easily check or set various communications parameters, such as the baud rate, the number of data bits, etc. Once control is transferred to this menu, it remains there until you exit to the Main menu with the X option.

The O option selects the OPTION MENU. From this menu, you can select the "trigger characters", as well as several other options. Once control is transferred to this menu, it remains there until you exit to the Main menu with the X option.

The U option selects the TEXT FILE UPLOAD MENU. From this menu, you can initiate the uploading of a text file to the remote system, or select various options concerning how this is to be done. Once control is transferred to this menu, it remains there until you exit to the Main menu with the X option, with the exception of the "upload" function itself, which automatically returns control to the transparent link once the upload is complete.

The D option selects the TEXT FILE DOWNLOAD MENU. From this menu, you can turn the text file capture mode on or off, control

whether "flow control" (XON/XOFF handshaking) is used, and if so, what characters are used to start and stop the flow of data from the remote system. Once control is transferred to this menu, it remains there until you exit to the Main menu with the X option.

The B option selects the BINARY FILE TRANSFER MENU. From this menu you can initiate a transfer of any CP/M file (including .COM files, etc.) to or from another CP/M system running MITE (or various other intelligent terminal programs) or an RCPM system. You can also select the protocol to be used. Once control has been transferred to this menu, it remains there until you exit to the Main menu with the X command. As before, once the actual Send or Receive functions have completed, control is automatically returned to the transparent link mode.

The M option selects the MACRO STRING DEFINITION MENU. From this menu, the user can view or change any of the ten macro strings which may be invoked via the "macro trigger" character. Once control has been transferred to this menu, it remains there until you exit to the Main menu with the X command.

The C option selects the SYSTEM COMMAND PROCESSOR, from which the user can issue a number of commands similar to those available in the CP/M "Console Command Processor" (i.e. command mode). These include such things as the DIR, ERA, and TYPE commands. Help is available by typing a "?". Once control has been transferred to this menu, it remains there until you exit to the Main menu with the X option, or enter an "empty" command line (immediate CR after prompt).

The F option selects the UNWANTED CHARACTER FILTER MENU. From this menu, you can view and modify up to 10 ASCII characters that should be discarded immediately upon receipt. They will not be displayed to the console or saved in memory. The first two of these characters default to 7FH (DEL) and 1AH (Ctrl-Z). The NULL character (00H) is automatically discarded.

The X option allows you to exit to CP/M. You will be asked to confirm this action before the exit is done:

"Are you sure (Y/N)? "

If you really do wish to exit at this time, type a "Y" or "y". Any other response will return control to the Main menu. If you do elect to exit, and a capture file is currently open, MITE will automatically close it for you, and inform you of this operation with:

"Capture Complete. Now closing file d:fn.ft"

If the carrier is still present, MITE will inform you of this with the following message:

"Warning... Carrier still present. Hangup (Y/N)? "

If you are finished with this session, reply with anything starting with "Y" or "y", which will cause MITE to hang up before exiting. If you merely wish to return to CP/M temporarily, and plan to resume this link, reply with anything else (typically "N" or "n").

The Parameter Menu

```
M.I.T.E. v2.x - Copyright (c) 1982, Mycroft Labs Inc.
OFFLINE. Bytes Captured =     0/32267. Capture = OFF.
Site ID = Source Telecomputing Corporation

PARAMETER MENU

     B - Baud Rate                  =   300
     D - Data Bits                  = 7
     P - Parity                     = EVEN
     S - Stop Bits                  = 1

     R - Role (ANS/ORG)             = ORG
     M - Mode (Duplex)              = FULL
     N - Phone Number               = 224-6824

     X - Exit to Main menu

Enter Option: _
```

On most of the above options the current value of the parameter is displayed to the right of option description. If the value is changed it is immediately updated on the menu display. Some of the options prompt the user for input (e.g. Baud). Others merely toggle between two or three states when selected (e.g. Mode). This approach makes it easy to determine the current setting of all parameters at a glance, and gives immediate feedback when they are being changed. Note that with all parameters, options, macro strings, etc. any change(s) you make will stay in effect only until you exit to CP/M, unless you use the S option on the Main menu to make the change(s) permanent.

The B option allows you to select a new baud rate. You will be prompted as follows:

"Enter New Value: "

This rate may be entered in any base, with post radix, the default (and normal) base being decimal. If an illegal value is entered (one not supported by your implementation and/or hardware), the message "Illegal Value" will be displayed, and the rate will remain unchanged. If an empty line (immediate <cr>) is entered, the rate will likewise remain unchanged. The default value is 300.

The D option allows you to select the number of data bits in each character. This option toggles between the values 7 and 8. Most timesharing systems use 7 data bits, and most systems that support binary file transfers use 8. The default value is 7.

The P option allows you to select the parity of each character. This option toggles between the values NONE, ODD and EVEN. Most systems that use 7 data bits will use EVEN parity, while most systems that use 8 data bits will use NONE. The default value is EVEN.

The S option allows you to select the number of stop bits on each character. This option toggles between the values 1 and 2. Most 110 baud (and slower) systems use 2 stop bits, virtually all other systems use 1. The default value is 1.

The R option allows you to select the "role" that MITE will play in a connection, the choices being ORG (originate) and ANS (answer). If you are dialing into another system, you should select the ORG role. If someone else is going to be dialing into you, you should select the ANS role. Note that not all hardware will support the answer role. The default value is ORG.

The M option allows you to select the mode (or duplex) of the transmission. It toggles between the values FULL and HALF. When running in FULL duplex, it is up to the remote system to echo any characters typed on the local keyboard back to the local display. In HALF duplex, it is up to the local system to perform this echo function. If you are getting NO echoes of characters you type, you are probably running FULL duplex on a HALF duplex system. If you are getting TWO echoes for every character you type, you are probably running HALF duplex on a FULL duplex system. Most on-line systems these days use FULL duplex. The default value is FULL.

The N option allows the user to specify the phone number of the remote site. Only numeric digits will be dialed. An asterisk may be used at any point to cause a one second delay. All other characters will be ignored, and may be used as desired to make the phone number more readable. On SMARTMODEM versions, a leading P will force Pulse dialing, and a leading T will force touch tone dialing. For more details on the SMARTMODEM, consult the user's manual supplied with each unit. To enter a blank phone number, enter at least one blank character. If the phone number field is blank, the dialing procedure will be skipped.

The X option allows control to return to the Main menu.

The Option Menu

```
M.I.T.E. v2.x - Copyright (c) 1982, Mycroft Labs Inc.
OFFLINE. Bytes Captured =     0/32267. Capture = OFF.
Site ID = Source Telecomputing Corporation

OPTION MENU

     E - Escape Trigger Char        = 0AH = ^J
     M - Macro Trigger Char         = 1BH = ^[
     B - Break Trigger Char         = 02H = ^B
     K - Command Trigger Char       = 0BH = ^K
     R - Remote Trigger Char        = 12H = ^R

     C - Caps Lock                  = OFF
     L - Auto LF after CR           = OFF
     T - TWX Mode                   = OFF
     D - Direct Connect Mode        = OFF

     X - Exit to Main menu

Enter Option: _
```

The E option allows the user to specify the "escape trigger" character that allows control to be transferred from the link back to the Main menu. Any time this user-specified character is typed (while in link mode), the screen will be erased and the Main menu will be displayed. From this point it is possible to initiate various functions, exit to CP/M, return to the link, or go to other menus. This character should be specified to be something not required on the remote system. The noraml default value is 1BH (27 decimal), which is the ASCII ESC character. When this option is specified, the user will be prompted for a new ASCII character, which may be entered in any of three ways:

1) numeric value in any base with post radix (e.g. 27, 1BH, 00011011B, etc.) Note: the first character must be a decimal digit 0 to 9

2) as a control code by entering a carat (^) followed by the character you are taking the "control" of (e.g. ^C for control-C)

3) by entering an ASCII control code directly (e.g. the ESC key).

Note that with the third method, certain characters cannot be entered due to the fact that CP/M processes them. These include Control-P, Control-M (CR), Control-J (LF), Control-H (BS), Control-X, Control-U, and Control-R. Note that both the hexadecimal value and the ASCII representation of the character

are displayed in the menu. Control codes are represented as an upper case alpha character preceded by a carat (e.g. ^F for Control-F). Since many function keys use <esc> as the first character of their output sequence (which means <esc> should be reserved for the "macro trigger character"), the Line Feed character (LF, 0AH, ^J) is a good alternative.

The M option allows you to specify the "macro trigger" character. Anytime this user-specified character is typed while in the link mode, a second character will be read. If it is a digit in the range 0 to 9, the corresponding macro string will be sent to the remote site as if it were coming from the keyboard. As with the "escape trigger" character, it should be specified to be something not normally required for use on the remote site. A new value may be specified in the same manner as with the "escape trigger" character described above. The normal default value is 0CH (Control-L). If function keys are to be used, the default value is usually 1BH (ESC). Note that if the character selected is required by the remote site, it may be sent by typing it twice.

The B option allows you to specify the "break trigger" character. Anytime this user-specified character is typed while in the link mode, a communications line BREAK function (SPACE condition for 150 milliseconds) will be performed. Note that this is NOT a real character, and is rather out-dated, but is still required by certain older computer systems (mostly IBM).

The K option allows you to specify the "command trigger" character. Anytime this user-specified character is typed while in the link mode, you will prompted with "Local Command?". For further information, see the Introduction under "Local Commands". This function is disabled when the value 00H is selected. The recommended value is ^K (0BH).

The R option allows you to specify the "remote command trigger" character. Anytime this user-specified character is typed on the ORIGINATE system while in the link mode, the ANSWER system will prompt you with "Remote Command?", in response to which the ORIGINATE user can issue "local" commands on the ANSWER system. This allows an ANSWER system to be put online in an unattended mode, so that the ORIGINATE user can control both ends of a file transfer, check directory space, etc, all from the ORIGINATE end. Note that the remote trigger character is determined by the ANSWER system operator, and is listed (if active, i.e. non-zero) when an incoming call is answered, as part of the initial greeting. Output of "local" commands invoked via the remote trigger character actually go to both consoles, and as a matter of convieniece, may be invoked and/or entered from either console (useful for training purposes).

The C option allows you to select an automatic upper case conversion in both directions. This affects only alphabetic characters, and serves the same function as a CAPS LOCK key. The default value is OFF. Each time this option is selected, the

value toggles between ON and OFF.

The L option allows you to select an Automatic LF (Line Feed) character to be displayed on the local console anytime a CR (Carriage Return) character is received. Some timesharing systems send only a CR at the end of each line, whereas most CP/M console terminals require both a CR and a LF to advance to the next line. The default value is OFF. Each time this option is selected, the value toggles between ON and OFF.

The T option allows you to enable or disable the Western Union TWX compatibility mode. For further information, see the comments in the Introduction, under "General".

The D option allows you to go into direct connect mode. This mode is used to "fool" the computer into thinking that a modem carrier signal has been received. This option should be used when connecting to computers together without the use of a modem. Another use would be for direct communications with a "smart modem". In most cases this option should remain off.

The X option allows you to return control to the Main menu.

The Text File Upload Menu

```
M.I.T.E. v2.x - Copyright (c) 1982, Mycroft Labs Inc.
OFFLINE. Bytes Captured =     0/32267. Capture = OFF.
Site ID = Source Telecomputing Corporation

TEXT FILE UPLOAD MENU

     U - Upload Text File

     D - Interchar. Delay          = 000
     E - Await Char. Echo          = OFF
     H - CR/LF Handshaking         = ON
     T - Turnaround Char.          = 00H = ^@
     G - Garbage Char. Count       = 000

     X - Exit to Main menu

Enter Option: _
```

The U option allows you to "upload" (send) a text file to the remote system, as if it were coming from the keyboard of the local console. When this option is selected, the user will be prompted for a filename, which should be entered in the "d:fn.ft" format. If the specified file is found, it will be sent to the remote system, using the conventions selected by the other options on this menu, as described below. Once the file has been sent, the communications link will be resumed automatically. If an XOFF character (13H, or ^S) is received from the remote system, MITE will pause until an XON (11H or ^Q) is received before continuing to transmit the file. This will prevent many systems from losing data in such situations. An upload can be aborted at any time by typing an ESC on the console keyboard.

The D option allows you to specify an "intercharacter delay" of 0 to 255 milliseconds. This allows you to slow down the outgoing text to the point that a "slow" remote system can accept it. This function works in both FULL and HALF duplex. It is the only way to slow down text in HALF duplex. Note that the time starts with the actual transmission of the character, not counting the transmission time at whatever baud rate you are running. If the actual delay time is less than a single character time, there will be no effect. For example, at 300 baud, each character takes about 33 milliseconds to transmit, so values below 33 will have no effect. A value of 100 will result in about 10 characters per second being sent. Note that this option does NOT affect baud rate, it merely inserts a variable length delay between characters which are being sent at the normal baud rate. When this option is selected, you will be prompted for a new value as with baud rate. The default value is 0.

The E option allows you to enable or disable a "wait for character echo" mode. When this mode is enabled, MITE will wait for each character that it sends to be echoed back by the remote system before it sends the next character. This insures that data will not be lost even on the "slowest" remote systems, but the overall through-put is much lower (usually about half) than the rate without the "wait-for-echo" enabled. When this option is enabled, MITE will compare each echoed character it receives against the character it sent, and list the number of characters that didn't match at the end of the transmission (nnnnn Compare Errors). Each time this option is selected, the value will toggle between ON and OFF. The default value is OFF.

The H option allows you to select the end of line (CR/LF) handshaking mode. When this mode is enabled, any time a CR (Carriage Return) is sent, MITE will wait until the remote end sends back a LF (Line Feed) before sending the next line. This is required on most timesharing systems, as they are usually "deaf" during this interval. If the user wishes to communicate with a remote printer, or simple datacomm utility (such as those found on many commercial word processors), it may be necessary to turn this option OFF, as the remote site in these cases will not send a LF in response to a CR. Each time this option is selected, the value will toggle beteen ON and OFF. The default value is ON.

The T option allows you to specify a "Turnaround" character. When set to 00H, this option is disabled. When set to any other value, at the end of each line, MITE will wait until it sees the specified character before it begins to transmit the next line. This allows the user to upload text to remote systems that prompt with a question mark (?), etc. When this option is selected, you will be prompted to enter a new ASCII character. The default value is 0.

The G option allows you to set the "Garbage Character Count". This refers to the number of characters that MITE will read and discard at the end of each line before starting to send the next line. Each such "garbage" character has a .5 second timeout. This feature allows the user to upload text to remote systems that send nulls, line numbers, or other extraneous characters at the start of each line before they are ready to receive the new line. When this option is selected, you will be prompted to enter a new value. The default value is 0.

Note that the H, T and G options can be used in combination, in which case the H option is processed first, then the T option, then the G option. This would allow the user to select a mode such as "At the end of each line, wait for a '?', then ignore the next 2 characters with a .5 sec timeout on each, THEN send the next line". Between the various options, it should be possible to upload text to virtually ANY online system.

The X option allows you to return control to the Main menu.

The Text File Download Menu

```
M.I.T.E. v2.x - Copyright (c) 1982, Mycroft Labs Inc.
OFFLINE. Bytes Captured =     0/32267. Capture = OFF.
Site ID = Source Telecomputing Corporation

TEXT FILE DOWNLOAD MENU

     C - Capture                    = OFF
     W - Write Captured Data
     R - Reset Capture Buffer
     P - Printer Echo               = OFF
     T - Type Capture Buffer

     F - Flow Control               = OFF
     Q - Flow Start Character       = 11H = ^Q
     S - Flow Start Character       = 13H = ^S

     X - Exit to Main menu

Enter Option: _
```

The C option allows you to turn the text capture mode ON or OFF. The first time capture mode is enabled (or the first time after a Write operation), the user will be prompted for a filename:

"Enter Filename: "

This filename should be entered in the "d:fn.ft" form (e.g. FRED.TXT). Note that it is possible to specify any CP/M logical device (LST:, PUN:, RDR: or CON:) in addition to disk file names. If the specified file already exists, you will be notified of this fact, and asked if you wish to overwrite it. Note that if you later exit to CP/M without writing the captured data to disk with the W option, MITE will automatically write it for you at that time. Each time this option is selected, the value will toggle between ON and OFF. The initial value is OFF.

The W option is used to write any captured data to the file that was originally specified when capture mode was first enabled. If flow control is enabled, this may actually be the last (partial) buffer. When the write is complete, MITE will remind the user of which file was being used:

"Capture Complete. Now closing file d:fn.ft"

This option will set capture mode to OFF, and reset the capture buffer to empty. When this option is selected, the write operation is performed at that time.

The R option allows you to reset the capture buffer to empty, which is useful only when flow control is disabled.

The P option allows you to turn ON or OFF a "printer echo" function. Anytime the printer echo is ON, any character that is written to the console during a link will also be written to the list device. In order for this to function properly, the list device must operate at an effective baud rate that is higher than that of the communications link. For slower printers, it may be necessary to use the "capture to LST:" mechanism, or to capture the data to a disk file for later printing. Each time this option is selected, the value toggles between ON and OFF. The intial value is OFF.

The T option allows you to type the current contents of the capture buffer to the console. The listing will pause every 23 lines (and at the end of file), until you hit a <CR> to continue, or <ESC> to abort the listing. Control-S can be used to pause the listing at any time. Note that this feature is normally of use only when Flow Control is not enabled (otherwise you can only type the last "partial buffer").

The F option allows you to select whether "flow control" handshaking is to be used while capturing text. This typically refers to XON/XOFF (control-S/control-Q) characters. Many systems will pause when you type control-S, and start back up when you type control-Q. If flow control is enabled, MITE will attempt to use this mechanism to cause the remote system to pause while it writes to disk the data it has captured since the last pause. This is normally done every 2048 bytes. When the flow stop character is sent out, MITE will wait until a full second has elapsed since the last character has arrived before writing to disk. This is necessary since some systems have characters "in the pipe" when the flow stop character is read. Each time this option is selected, the value will toggle between ON and OFF. The default value is OFF. If the remote system does support this convention, we highly recommend that you take advantage of it.

The Q option allows you to specify the flow start character as described under the F option. Most systems that have this feature use control-Q. When this option is selected, you will be prompted to enter a new ASCII character. This value may be specified in the same three ways that the "escape" character may be defined. The default value is ^Q.

The S option allows you to specify the flow stop character as described under the F option. Most systems that have this feature use control-S. When this option is selected, you will be prompted to enter a new ASCII character. This value may be specified in the same three ways that the "escape" character may be defined. The default value is ^S.

The X option allows you to return control to the Main menu.

The Binary File Transfer Menu

```
M.I.T.E. v2.x - Copyright (c) 1982, Mycroft Labs Inc.
OFFLINE. Bytes Captured =     0/32267. Capture = OFF.
Site ID = Source Telecomputing Corporation

BINARY FILE TRANSFER MENU

     P - Protocol                    = XMODEM

     S - Send File and Return to Link
     R - Receive File and Return to Link

     X - Exit to Main menu

Enter option: _
```

The P option allows you to select the protocol to be used in the transfer of binary files from one system to another. The protocols currently supported are:

MITE	The native MITE protocol (as used in the Mycroft Labs SEND/RECV utilities, see the August 1982 Dr. Dobbs Journal). This protocol supports "afn" transfers (e.g. all files of type .ASM). No file name is required on the receive end.
XMODEM	The protocol used on most RCPM systems. Compatible with MODEM7, XMODEM, MODEM80, various other intelligent terminal programs.
CLINK	The original Mycroft Labs binary file protocol. Compatible with CLINK and CROSSTALK.
HAYES	The Hayes Terminal Program Verification protocol.
IBMPC	The IBM Async Support Package text file protocol. Note that this is not strictly a binary protocol, as only text files can be sent, and no error checking or retransmission is done.
TEXT	A simple ASCII text file protocol that can interact with a high-level language program on a mini or mainframe to transfer text files with error checking and recovery.

When the P option is selected, the following display will appear, at which time you may enter one of the characters listed, or a CR (carriage return) to leave the protocol unchanged:

Current protocol is: XXXXXX

C - CLINK / XTALK	**H - HAYES**
I - IBMPC Async	**M - MITE Multi-file**
T - TEXT (mainframes)	**X - XMODEM**

Enter new protocol or CR for no change:_

Be sure that the system that you plan to communicate with supports at least one of these protocols. When this option is selected, it will toggle between the supported protocols. The default value is XMODEM.

The S option allows a file to be sent using the currently selected protocol. When this option is selected, you will be prompted for a filename, which should be entered in the "d:fn.ft" format. At this time, MITE will start sending the specified file to the remote system. A period (".") will be displayed on the console for each block sent and acknowledged correct. An "R" will be displayed for each block that was rejected. When the file has been completely sent, MITE will display the message: "File Sent" and the transparent link will be automatically resumed.

The R option allows a file to be received using the currently selected protocol. When this option is selected, you will be prompted for a filename, which should be entered in the "d:fn.ft" format. At this time, MITE will start receiving the specified file from the remote system. A period (".") will be displayed on the console for each block received correctly. An "R" will be displayed for each block that is received in error. When the file has been completely received, MITE will display the message "File Received" and the transparent link will be automatically resumed.

The X option returns control to the Main menu.

The Macro String Definition Menu

```
M.I.T.E. v2.x - Copyright (c) 1982, Mycroft Labs Inc.
OFFLINE. Bytes Captured =     0/32267. Capture = OFF.
Site ID = Source Telecomputing Corporation

MACRO STRING DEFINITION MENU

     0: mailck^M
     1: mail read^M
     2: post scan cp/m^M
     3: off^M
     4:
     5:
     6:
     7:
     8:
     9: ^M^W^M^E^T=d1^M^T@c 30128^M^T>id tcm495 xxxxx^M

     X - Exit to Main menu

Enter Option or ? for help: _
```

This menu allows you to view and/or define up to 10 prestored macro strings (each of which can be up to 62 characters in length). These strings are saved and loaded along with the parameters in the .PAR file. Typical uses for these would be for semi or fully automatic login or favorite commands. The example shown above might be used on the Source. To define the 'n-th' string, type the number of the desired string (0 to 9). You will be prompted for a new string, which will then be displayed in the menu following the string number. To exit from this menu, use the X option. Any printable ASCII characters can be entered directly. To enter control codes (such as CR, LF, control-X, control-H, etc.), enter a carat (^) followed by the printable character corresponding to it. The following table will help in determining which characters to use for various control codes:

NUL	00H	^@	BS	08H	^H	DLE	10H	^P	CAN	18H	^X
SOH	01H	^A	HT	09H	^I	DC1	11H	^Q	EM	19H	^Y
STX	02H	^B	LF	0AH	^J	DC2	12H	^R	SUB	1AH	^Z
ETX	03H	^C	VT	0BH	^K	DC3	13H	^S	ESC	1BH	^[
EOT	04H	^D	FF	0CH	^L	DC4	14H	^T	FS	1CH	^\
ENQ	05H	^E	CR	0DH	^M	NAK	15H	^U	GS	1DH	^]
ACK	06H	^F	SO	0EH	^N	SYN	16H	^V	RS	1EH	^^
BEL	07H	^G	SI	0FH	^O	ETB	17H	^W	US	1FH	^_

For example, to enter a CR (Carriage Return, or 0DH) use the code ^M. Note that two carats in a row (^^) will be interpreted as a

single carat. This means that the RS control code cannot be entered in this manner.

There are several special control codes that may be used to automate the login process on many remote systems. The example in the menu above is for the Source. The characters currently available are as follows:

```
^M   Carriage Return
^E   Turn on wait-for-echo mode
^N   No echo wait (default)
^W   Wait until no characters received for 1 full second
^Tn  Trap on the ASCII character "n"
```

If macro string 9 is defined, it will be sent automatically the first time a connection is established on an outgoing call. The console keyboard is also active when a macro string is being sent, and all characters except ESC will be sent to the remote system as usual. The ESC character is used to abort the current string. NOTE: due to variations in the way that typical remote systems login, none of the "auto login" features of currently available programs will work on all systems, or even every time on a given system.

As an example of an auto login, the following string may be used to logon to the Source via Telenet:

```
^M^W^M^E^T=d1^M^T@c 30128^M^T>id tcm495 xxxxx^M
```

When invoked, this string performs the following functions:

```
^M          First CR to get attention (not echoed)
^W          Wait for 1 second
^M          Second CR to get attention (not echoed)
^E          Turn on wait-for-echo mode
^T=         Input and display characters until "=" is seen
d1^M        Response to "Terminal=" question on Telenet
^T@         Input and display characters until "@" is seen
c 30128^M   Response to "@" prompt on Telenet
^T>         Input and display characters until ">" is seen
id ...^M    Source logon command
```

The "wait-for-echo" mode forces MITE to wait for the echo of each character that it sends before proceeding to the next. This mode defaults to OFF at the start of each string. Some systems will lose characters if this wait is not enabled. However, many systems do not echo initial CR characters, passwords, etc., hence there must be a means of turning this mode on and off on a character by character basis.

To invoke a macro string during a link, enter the "macro trigger character" (see Option Menu), followed by the 'n-th' index character as set up for your terminal (see installation notes). This mechanism allows the installer to make use of the function keys of most terminals to invoke the strings. For

example, on a Zenith Z19/Z89 keyboard, the function keys generate the following codes:

BLUE:	ESC P	F2:	ESC T
RED:	ESC Q	F3:	ESC U
WHITE:	ESC R	F4:	ESC V
F1:	ESC S	F5:	ESC W

Hence in that version, the "index characters" are set to:

"STUVWPQR89"

This means that the ten macro strings can be invoked by the following keys:

String 0: F1	String 5: BLUE
String 1: F2	String 6: RED
String 2: F3	String 7: WHITE
String 3: F4	String 8: ESC 8 (answerback)
String 4: F5	String 9: ESC 9 (auto logon)

Anytime the "macro trigger" character is typed in the link mode, if the next character is one of the ten installed "index characters", then the corresponding macro string is sent, as if it were coming from the keyboard. If the character following the "trigger character" is NOT one of these index characters, then that second character is sent through normally. One ramification of this is that if the remote system needs the "macro trigger" character itself, it can be sent by typing it twice. Note that this does not work on the "escape trigger" character. See the Installation Guide for further tips on taking advantage of existing function keys.

Note that macro string 8 is used as the "answerback" string for use in the TWX compatibility mode.

Unwanted Character Filter

```
M.I.T.E. v2.x - Copyright (c) 1982, Mycroft Labs Inc.
OFFLINE. Bytes Captured =     0/32267. Capture = OFF.
Site ID = Source Telecomputing Corporation

Unwanted Character Filter Definition Menu

        0: 7FH =
        1: 1AH = ^Z
        2: 00H = ^@
        3: 00H = ^@
        4: 00H = ^@
        5: 00H = ^@
        6: 00H = ^@
        7: 00H = ^@
        8: 00H = ^@
        9: 00H = ^@

        X - Exit to Main Menu

Enter Option: -
```

This sub-menu gives the user the capability of ignoring certain characters received from the remote system. This may be useful in the case of the micro communicating with a mainframe that uses the delete character (7FH) as a pad character. Many microcomputers will do a character delete from the screen when this is received. In order to prevent this, the user would select the 7FH as a character to ignore from the remote system. (The 7FH character is a standard character to be ignored.)

Three methods of entering the value is allowed. The ASCII value of the character may be entered in decimal or in hex. The control character may be entered by pressing the control key along with the appropriate character. In some cases CP/M will trap this and not return it to MITE which leads to the third method of entry; enter "^" followed by the character (e.g. "^M" to indicate a carriage return, 0DH).

The System Command Processor

```
M.I.T.E. v2.x - Copyright (c) 1982, Mycroft Labs Inc.
OFFLINE. Bytes Captured =      0/32267. Capture = OFF
Site ID = Source Telecomputing Corporation

SYSTEM COMMAND PROCESSOR

Enter command, HELP or ? for details: _
```

Available commands:

COPY new=old	copy one file to another
DIR afnx	list directory of files
DRIVE d:	select new drive as default
ERA afnx	erase file(s)
HELP	print list of legal commands
LIST d:ufn	list file to printer
REN new=old	rename "old" file to "new"
RESET	make all drives read/write
SET afnx $att	give file(s) new attribute
SIZE afnx	list size of file(s) in k bytes
SPACE d:	show space available on d:
TYPE d:ufn	list text file on console
USER n	select new user number

The "system command processor" understands a number of commands similar to the CP/M "direct" commands (e.g. DIR, ERA, TYPE, etc.), as opposed to transient commands (e.g. STAT, ED, etc.). When this option is selected, any number of these commands may be entered, one at a time. Once the command has been performed, you will be prompted as follows:

"Enter <CR> to enter another command, X to exit to Main menu:"

which is self-explanatory.

The parameters for many of the above commands can be specified in the same way as the corresponding commands in the CCP, but in MITE, they are actually more generalized. Anywhere you can specify an 'afn' (ambiguous filename), in MITE you can specify an 'afnx' (ambiguous filename expression). This is a list of one or more ambiguous filenames seperated with + or - operators, for set inclusion or exclusion. A '+' means to include any of the filenames covered by the following afn that are not already in the overall list, while a '-' means to exclude any of the filenames covered by the following afn that ARE currently in the overall list. Note that the drive name is not significant in the comparisons.

As an example, the afnx "*.ASM+*.SRC-F*.*" would mean all files of type '.ASM' and all files of type '.SRC' except those beginning with 'F'. It is also possible to use an expression like "A:*.ASM-B:*.ASM" which means all files on drive A: of type .ASM except those that also occur on drive B:.

If you need more detailed information on the way these commands are described, type a question mark (?) in place of a command, and a entire page of explanation will be displayed, The commands available are as follows:

COPY new=old	Simple file to file copy, uses syntax similar to PIP (e.g. COPY DEMO=TEST.DAT to make a copy of TEST.DAT on the file DEMO).
DIR d:afn	List the names of all of files on disk 'd' that satisfy the ambiguous filename 'afn'. For example, 'DIR *.ASM' will list the names of all files of type 'ASM'. If the drive (d:) is not specified, the current drive is assumed. Names of R/W files will be preceded with the standard ':', while names of R/O files will be preceded with a '>'. SYS files are not listed.
DRIVE d:	Equivalent to 'd:' by itself in CP/M command mode. This makes drive 'd' the new default (or current) drive.
ERA afnx [q]	Erase all files on disk 'd' that satisfy the ambiguous filename 'afn'. For example, 'ERA *.ASM' will erase all files of type 'ASM'. If the drive (d:) is not specified, the current drive is assumed. If a second parameter of Q is specified (ERA *.ASM Q) then you will be asked YES or NO on each file before it is deleted.
HELP	Print list of available commands.
LIST ufn	List specified file to printer.
REN new=old	Give an existing file ("old") a new name ("new"). Note that both "old" and "new" must be unambiguous filenames. For example, to change the name of the file "FRED.ASM" to "JOE.MAC", enter "REN JOE.MAC=FRED.ASM".
RESET	Make all drives Read/Write again. Equivalent to a control-C (^C) in CP/M command mode. Should be done anytime a new disk is mounted.

SET d:afn $att — Give all files on disk 'd' that satisfy the ambiguous filename 'afn' the new attribute 'att'. If the drive (d:) is not specified, the current drive is assumed. Possible new attributes are:

$DIR - make files visible in directory
$SYS - make files invisible in directory
$R/W - make files Read/Write
$R/O - make files Read/Only

For example, to make all files of type 'COM' on the current disk Read/Only, use the command: 'SET *.COM $R/O'.

SIZE afnx — List size of specified file(s) in K bytes, then list total size of those listed. This can be helpful for estimating transmission time for a file.

SPACE d: — List space available on specified drive. If no drive is specified, the current drive is assumed.

TYPE ufn — List specified file to the console. Control-S may be used to pause (once paused, any character will restart listing), or any other character will abort listing.

USER n — Set user number (0 to 15).

Note that all filename wildcard conventions (* and ?) used in CP/M command mode are supported.

Creating Parameter Files

Once a parameter file has been created for a given site, it is quite easy to dial into that site and begin using it, without having to look up a phone number, password, etc. To create such a file, the typical procedure would be as follows:

1. Bring up MITE either with all defaults, or with another parameter file if one exists that is close to what you want (say, identical except for phone number).

2. Bring up the Parameter Menu and change any incorrect values. The most common things to set here are the phone number, baud rate, parity, number of data bits, number of stop bits and mode (full or half duplex). Most online systems use 300 baud, 7 data bits, EVEN parity, 1 stop bit and full duplex.

3. Bring up the Option Menu and change any incorrect values. The most common things to change here would be trigger characters that might conflict with characters needed at the remote site.

4. If you plan to send text files TO the remote system, bring up the Text File Upload Menu and change any incorrect values. Typically the values on this menu would be set based on trial and error once a connection had been established, but some options may be obvious on certain systems.

5. If you plan to capture text files FROM the remote system, bring up the Text File Download Menu and change any incorrect values. The most common things to change here would be whether flow control is supported or not, and what the start and stop characters are, if so.

6. If binary file transfers are anticipated, bring up the Binary File Transfer Menu and select the protocol to be used. See the Binary File Menu description for a list of programs compatible with the various protocols MITE supports.

7. If any pre-stored macro strings (such as logons, passwords, favorite commands, etc.) are desired, bring up the Macro String Definition Menu and define them now. If you do enter any passwords here, be careful not to let this file fall into the wrong hands. Note that most strings should be terminated with a ^M to cause a carriage return to be sent at the end of the string.

8. Return to the Main menu and select the S option. The name specifed in response to the 'Enter Filename: ' prompt may be specified on the command line (or in response to the L option) in future sessions.

Uploading and Downloading Text Files

At any point, you can turn the text capture mode on (see the Text File Download Menu). The first time you do this, you will be asked for a filename. This is the name of the file to which you want the captured data to be written. Until you terminate this mode with the W command, or by exiting to CP/M, any character written to the screen while in link mode will be captured on that file. It is possible to temporarily disable the capture mode by turning it OFF, then back ON later. This mechanism can be used to make a permanent record of an entire session. If flow control is enabled, you may notice short pauses every 2,000 characters or so. Do not let these worry you - this is MITE taking a second to write some more data to disk. Note that the ASCII characters 00 (NUL), 1AH (Control-Z or logical EOF) and 7FH (DEL) are automatically discarded.

It is also quite easy to send text files to the remote system as if they were being typed on your keyboard. This can save a great deal of time on long distance calls, or on systems that charge on a "connect-time" basis. It can also allow you to leave nicely formatted messages (by creating them with a word processor before making the connection). To do this, put the remote system into a mode where it would normally expect you to start typing away on a message or text file (e.g. insert mode in an editor), then go to the Text File Upload Menu and issue the U command. You will be asked for a filename. This is the name of the file that you wish to send. DO NOT SPECIFY A BINARY FILE. It would probably confuse the remote system rather badly. You have several options concerning how the file is to be uploaded. See the Text File Upload Menu description for details.

A Typical Session on The Source

A typical session might proceed as follows. In this session, we will dial into the Source and upload a message into its Electronic Mail system. We will also capture the entire session for future reference.

1. A disk containing MITE.COM is mounted on the system in Read/Write mode.

2. MITE is executed, with a parameter file:

```
A>mite source
```

3. The G option is selected from the main menu. This causes MITE to dial the prestored phone number (assuming this is an auto-dial version). Once carrier is detected, MITE will inform you of this fact, and you will be in transparent link mode.

4. At this point, the login is performed automatically, as described under the Macro String menu.

5. In this case, it is desired to keep a disk copy of the entire session, so capture mode will be turned on and left that way for the entire session. None of the other operations will affect this capturing adversely. To do this, the "escape trigger character" (in this case LF) is typed, which returns control to the main menu. The D option is selected, which brings up the Text File Download Menu. On this menu, the C option (Capture) is selected. Since this is the first time this option has been selected, a filename is asked for. Because the remote system supports flow control, an arbitrarily long session may be captured (assuming sufficient disk space). Once the filename is entered, the X option is used to return to the Main menu, where the G option resumes the transparent link.

6. A prestored command is invoked (with a single function key) to check for incoming mail (via the command MAILCK).

7. An outgoing message for someone has been previously created on a disk file with the ED text editor. The commands necessary to put the remote system into a mode where the text of the message would normally be entered manually are issued: "MAIL SEND XYZ123". At this point, the "escape trigger" character (still LF) is typed, which returns control to the Main menu. The U option is chosen, which brings up the Text File Upload Menu. From here the U option is chosen, which asks for a filename. The name of the previously created text file is entered. At this point, the contents of that file are sent to the remote system as if they were being typed on the keyboard. Once the entire file is sent, the transparent link is automatically resumed, and the message can be closed out and mailed.

8. In this example, that is all that was desired, so the system is logged off with another prestored macro string (OFF). On most systems, that would cause carrier to drop, which would automatically return control to the Main menu. On this system, the user is returned to TELENET without the carrier being dropped. The "escape trigger" character is used to return to the Main menu one last time. From here the X option is used to exit to CP/M. The first question asked is "Are you sure?". We are, so we type Y (for yes). Since we forgot to write the last buffer to disk with the W option on the Text File Download Menu, this is now done automatically, and the name of the file is printed to remind us. Since carrier is still present, you are reminded of this, and offered the option of hanging up at this time. Since we are finished, this option is selected, after which MITE exits to CP/M.

Sending and Receiving Binary Files

Not all files under CP/M are simple "text files". A text file is something that yields meaningful results when listed to the console with the "TYPE" command. An example of a "binary" file is a .COM executable object file, a .REL relocatable object file, Wordstar internal files, various data files created by programs like SuperCalc, etc. In order to transfer this kind of file, a more powerful transfer mechanism is needed. Note that simple text files are a special case of "binary file", and may also be transferred with this mechanism. If binary file transfer is possible, then that is the preferred mechanism, as retransmission of blocks is performed automatically on detection of errors. This greatly improves the chances of getting the file to the remote site intact. For the most part, binary file transfer is only practical with another CP/M system.

There are numerous "online" CP/M systems around the country that have a large number of public domain programs on them. These systems are referred to as "Remote console CP/M" (or RCPM) systems. They all have a program called XMODEM that supports file transfers in either direction, with error checking and automatic recovery. Any kind of file may be transferred using this program. In order to do this, though, you must have a program on your end that knows how to communicate with XMODEM. There is a simple file transfer program in the public domain called "MODEM7" that can do this, but it is rather clumsy to use, and even more so to get ahold of and to install on any particular system.

One of the nicer features of MITE is support for the file transfer protocol used by XMODEM. In order to use this, you must bring up the Binary File Transfer Menu and select the XMODEM protocol via the P option. Note that this protocol may also be used for file transfer between two MITE systems, as long as both systems have selected the XMODEM protocol. Most of these online systems have documentation on how to use XMODEM, but briefly, it is run by specifying either an "R" (to receive) or an "S" (to send) as the first parameter, and a filename as the second parameter (e.g. "XMODEM R FRED.ASM" to receive a file from you and write it to FRED.ASM on the RCPM system).

To Receive a file from an RCPM system onto your disk:

1. Login to the RCPM system like you would with any other online system. Few if any of these systems have passwords.

2. Start the XMODEM program (in Send mode) on the RCPM system: "XMODEM S fn.ft".

3. Escape from the transparent link and bring up the Binary File Transfer menu (option B). From there, select the R option (to Receive a binary file) and specify the filename that you want this file saved under.

To Send a binary file to an RCPM system from your disk:

1. Login to the RCPM system like you would with any other online system.

2. Start the XMODEM program (in Receive mode) on the RCPM system: "XMODEM R fn.ft".

3. Escape from the transparent link and bring up the Binary File Transfer menu (option B). From there, select the S option (to Send a binary file) and specify the name of the file you want to send.

In either case, a period will be printed on your console for each 128 byte block that is transferred successfully. An R will be printed for each block that had to be re-transmitted. Once the transfer is complete, control will automatically be returned to the transparent link.

To exchage files with a CLINK (or CROSSTALK) user, a similar sort of process is used. Sending files TO a CLINK system is especially easy, if the CLINK system has enabled a REMESC (Remote Escape character). In that case, everything can be done from your end. In any case, the CLINK protocol must be selected (via the P option on the Binary File Transfer menu).

To Receive a file from a CLINK system onto your disk:

1. Establish a connection with the CLINK system. One of you should be in ANSWER mode and the other in ORIGINATE mode. It doesn't really matter which end plays which role, so long as they are not the same.

2. Once the two users have agreed on the file to be sent, the MITE user should escape from the link and bring up his Binary File Transfer menu. From there, the R option should be specified, and the name of the file to which you want the received information to be written entered in response to the prompt.

3. At this time, the CLINK user should issue a local "SEND" command: "<esc> SEND fn.ft".

To Send a file to a CLINK system from your disk:

1. Establish a connection with the CLINK system as above.

2. Either have the CLINK user issue a local RECV command, or you can enter it for them if they have enabled the REMESC function. In either case, the command would be: "<esc> RECV fn.ft".

3. At this time, the MITE user should escape from the link, and bring up the Binary File Transfer menu. From there, the S option should be specified, and the name of the file you wish to send entered in repsonse to the prompt.

In either case, a period will be displayed for each block (typically 1024 bytes in length) transferred intact, and an R will be displayed for each block that had to be re-transmitted. Various other characters may be listed under certain circumstances, such as T for timeout or C for CRC error.

Setting Up an Unattended Answer System

With this version of MITE, it is possible to set up an unattended answer system for other users to dial into, so that they can send files to (or receive files from) your system. This can be quite useful for distribution of public domain software in clubs, etc. The user dialing into such a system can request a directory of files on any of your disks, erase files, type files, check file size, check space on the disk, check/set protocol, or start a file transfer in either direction.

All that is required is to set up a normal answer system, and then define a non-zero "remote trigger character" (R command on the OPTION menu). Obviously an auto-answer modem is required for this type of operation. The recommended value (and normal default value) is ctrl-R (^R, or 12H). Anytime there is a non-zero value for this character, its value will be displayed to any incoming user, along with the normal greeting.

Once someone dials into such a system, they can issue "local" commands on the ANSWER system by typing the remote trigger character. The phrase "Remote Command?" will be displayed on their system, at which time they may enter any "local" or "system" command, which will be performed on the ANSWER system. The results of that command will be displayed on BOTH consoles. When in doubt, the dialed-in user may use the HELP command.

Installation Notes

If you have a terminal and modem currently supported by Mycroft Labs, it is quite simple to install MITE, using our INSTALL program. The procedure for doing this is as follows:

1) Make a working copy of the distribution disk onto one of your own diskettes. Place distribution disk in a safe place so that you will have it in case you need it at a later time. DO NOT INSTALL ONTO DISTRIBUTION DISK.

2) Mount this new diskette in drive A: and run INSTALL.

3) Select terminal/computer type from the menu by selecting the appropriate letter.

4) If you wish to have a "default" phone number, enter it in response to the next question. The phone number entered at this time will be entered and used each time MITE is run (unless a parameter file is used). If left blank, no phone number will be used unless entered through the parameter menu. In most cases, a return should be pressed to leave the phone number blank.

5) The next five questions allow the user to select a different default escape character, macro trigger character, break character, local command character and remote command character. These may be changed to other characters, however it is suggested that the novice user simple press the return to allow the standard defaults to be used. This will provide consistency between this manual and the working program.

6) You will next be asked to select the computer/serial I/O port to be used. If your type computer/serial port is not specified in the list, contact Mycroft Labs or your nearest dealer for help installing your specific equipment.

7) The final question allows you to select the modem type to be used. This question specifies the type of auto-dialing to perform. If your modem is a manual dial type, select the manual option. If no modem is to be used, select direct connect. The direct connect option will allow computers to be connected directly to each other without modems. If this is desired, a special "reverse" cable may be required. Consult your nearest computer store for aid in doing this.

8) At this point, INSTALL will write the installed version of MITE to the file MITE.COM. If you want this file named otherwise, include an "O=fn" parameter on the command line (e.g. INSTALL O=MYMITE.COM). The default input file is MITE/U.COM. If for any reason this needs to be changed, include an "I=fn" parameter on the command line.

If you wish to install MITE for a terminal other than one supported by Mycroft Labs, you will need to use DDT or SID to

patch in certain hex values in the first page of code as described below. For information on using DDT and/or SID consult the appropriate Digital Research manuals, or contact a system programmer.

There are a number of storage locations near the beginning of MITE that the user can patch to customize MITE for their terminal. A "sequence" starts with the number of characters (e.g. 2) then that number of characters (e.g. 1BH,'E'). These locations are as follows:

010C:	screen erase sequence
0112:	ten "index" chars for patching into function keys
011C:	default phone number, terminated by 0 byte
0131:	default escape character
0132:	default "trigger" character
0138:	screen highlight ON sequence
013E:	screen highlight OFF sequence

As an example, for a Zenith Z19 terminal, these might be:

010C:	2,1BH,'E',0,0,0
0112:	'STUVWPQR89'
011C:	'nnn-nnnn ',0
0131:	0AH
0132:	1BH
0138:	2,1BH,'p',0,0,0
013E:	2,1BH,'q',0,0,0

Values for other terminals can usually be determined from the user manual supplied with the terminal.

If you wish to install MITE for a serial port/modem combination not currently supported by Mycroft Labs, contact us for access to the source code of the driver for the system that is most similar to your hardware. The procedure is quite similar to installing CP/M on new hardware, but a good deal easier.

Basically the process involves creating an absolute assembly language module (in 8080 or Z80 code) with a jump vector at the beginning of it, followed by a number of subroutines that perform functions like "check for carrier present", "set parity", "dial number", etc. There are 1024 bytes available for these routines.

Given access to the hardware in question, Mycroft Labs will contract to create this module for a one time charge of $200.00. If the user manages to successfully install MITE themselves, Mycroft Labs may well be interested in acquiring the module either through outright purchase or in exchange for Mycroft Labs software. Please contact us directly if you are interested in pursuing any of these alternatives.

Appendix A

Introduction to Data Communications

Basic Concepts - Information encoding

One of the fundamental concepts in the data communications field is that of "encoding" information from human-sensible form(s) into machine-sensible form, and "decoding" machine-sensible back into human-sensible. A teletype compatible terminal is a simple hardware device which is capable of performing both of these actions. There are several common means of encoding information so that a machine can "sense" it, almost all of which involve reducing said information to "binary" as a first step. Information theory tells us that in the general case, any information can be reduced (reversibly) in this manner. A single binary digit ("bit") can be represented by the presence or absence of a current, one of two possible voltages, one of two possible audio frequencies, etc. Likewise, there are several common conventions for associating a group of bits to represent a particular character, numeric value, machine instruction, etc.

From one point to another within a terminal (e.g. the link between the keyboard and the rest of the terminal), a simple encoding scheme would typically be used which is called "TTL level parallel". The TTL stands for "Transistor-Transistor Logic". Under the TTL encoding scheme, a logic "0" is represented by a voltage in the range 0 to 1.3 VDC, and a logic "1" is represented by a voltage in the range 3.6 to 5.0 VDC. The current involved is very small, hence TTL links must be physically very short (typically 2 to 3 feet at most). There are no widely adopted conventions concerning how to arrange the signals on a TTL link, or even which signals to have present.

From one device to another (e.g. terminal to modem) one would typically use either current loop (on older equipment) or EIA RS-232-C (on most all recent equipment). A current loop is implemented by physically breaking a wire loop (or equivalent) in which a 20 mA current is flowing. Here a logic "1" is represented by the presence of current (MARK) and a logic "0" by the absence of same (SPACE). Historically, a long chain of "half duplex" teletypes were hooked up "serially" on a single loop (often with the "return" wire being one rail of a train track). When any one terminal was typed on, the entire loop was "made" and "broken", hence the current was present and absent, in a binary representation of the character that was typed. Since all the printer mechanisms in the loop decoded the information present in the current and printed the character corresponding to whatever pattern(s) were detected, they all printed the character typed. Once again, no real standards exist concerning the type of connectors to use, hence moving a current loop device from one environment to another usually involves rewiring connectors.

As computer peripherals began to proliferate, and all of them had their own proprietary (and incompatible) interface

conventions, the Electronics Industries Association (EIA) decided to provide a generalized interface standard. If a given manufacturer stuck reasonably close to it, he stood an excellent chance of being able to "plug" his equipment directly into other such devices using standard, readily available cables and/or connectors. Although there are a bewildering variety of "subsets" of this standard (virtually no one ever implements the entire standard), one can usually connect two "RS-232 compatible" devices together with a minimum of hassle. The RS-232-C specification not only specifies the physical encoding scheme (logic "0" = +3 to +25 VDC, typically +12, logic "1" = -3 to -25 VDC, typically -12), it also specifies which pins of a "recommended" connector (DB25P/DB25S) are to carry which signals, and how those signals interact ("handshaking"). As somewhat higher currents are used than in TTL, a simple RS-232 link may be somewhat longer, typically up to 50 feet.

Often, one needs to connect two devices over even longer distances (possibly even over thousands of miles). As the switched telephone network is already there, and is "reasonably" inexpensive to use, engineers have designed devices which allow one to couple two RS-232 devices via a voice grade phone line, such that the information is encoded into one of two possible audio frequencies in the one to three kilohertz range, and back again. Since this process involves both "modulation" and "demodulation" of a carrier frequency, these devices are called "modems". Here the distance is limited only by the extent of the phone network. As in all other areas of data communications, there are a number of similar, yet incompatible, mechanisms for encoding information, each method optimized for some combination of bandwidth, signal quality, and cost.

Basic Concepts - Information Structure

The next level above representing individual bits in some machine sensible form is to impose some sort of structure on groups of such bits. Once again, there are several commonly used conventions, each optimized for a given set of parameters.

Returning to the example of the keyboard TTL link - due to the short length of the connecting cable, chances are that such a link would use a "byte parallel" interface. That is, the (in this case seven) bits that represent a given character would all be transmitted simultaneously over separate wires. By maintaining an order among the various wires, the character can be received intact by the rest of the terminal's circuitry.

Often, one wishes to reduce the required number of wires over a link to a minimum. An RS-232 link can be established with as few as three wires (two if information flows in only one direction). This is achieved by allowing multiple bits to "time-share" a single wire. Once again, there is more than one convention for doing this: asynchronous serial and synchronous serial. The connection between a terminal and its modem is typically RS-232 asynchronous serial.

Asynchronous serial involves sending information with precise timing between bits of a given character, but with arbitrarily long (including zero-length) intervals between characters. In order to allow this variable length interval, certain overhead information is required on each character. Normally, during an inter-character gap, the line is in the logic "1" (MARK) state. When a character starts, there is a single "start bit" which is always logic "0" (SPACE). Once the start bit has occurred, then the "data bits" occur, least significant bit first. One typically finds either 7 or 8 such data bits on recent equipment. Following these, one often finds a "parity bit", which is a simple error detection mechanism. If running with "even parity", this bit will be chosen so that the total number of bits (in the data bits + parity bit part of the character) is an even number. Following the parity bit (if it is present) are 1 or 2 "stop bits" (always logic "1" (MARK)). Stop bits simply insure that the line returns to the idle (MARK) state for at least one (or two) bit time(s), so that the hardware can re-cycle and be ready to detect the next start bit. This technique is commonly employed in situations where a human determines the timing between characters (such as in most terminals with keyboards). It is suitable for relatively low speed applications (1 to 1000 characes per second). A single inexpensive ($5.00) LSI chip is available to convert parallel data to or from asynchronous serial data, called a UART (Universal Asynchronous Receiver / Transmitter). Note that all connections to the UART are at TTL levels, hence TTL to RS232 and RS232 to TTL level shifters (such as 1488 and 1489) are usually found in conjunction with a UART.

Synchronous serial involves sending characters not only with precise timing between bits of a given character, but at regular intervals (no gap between characters), typically in relatively long (e.g. 1024 character) blocks. No overhead is required on each character, although parity is still sometimes used. However, certain overhead information is required at the start and end of each block. There are two main types of synchronous serial, namely "byte control protocols" (BCP), such as DDCMP (DEC's Digital Data Communications Message Protocol) and BISYNC (the older IBM standard); and "bit oriented protocols" (BOP) such as SDLC (Synchronous Data Link Control) and HDLC (High-level Data Link Control). A BCP can function with conventional byte-oriented hardware, but a BOP requires special bit-oriented hardware. The primary function of the overhead information is to achieve "synchronization", such that both ends agree where the boundaries between bytes are in the continuous stream of bits. Typically, this overhead information also supplies a certain amount of redundancy for sophisticated detection (or even correction) of errors. Typically used in relatively high speed applications (100 to 100000 characters per second). Note that since the timing is more critical (and must be sent along with the data), and the speed is generally higher, modems for synchronous serial data are typically more complex and expensive that those for asynchronous serial data.

Typically even higher levels of organization than this are imposed on synchronous serial data, up to the "message" level and beyond. This includes mechanisms to support acknowledgement of messages received intact, requests for retransmission of messages received with errors, end of transmission status, etc.

Any serial communications link can be set up in one of three modes: "simplex", "half duplex", or "full duplex". The term "echoplex" is often used to refer to a full duplex channel which is being used primarily in one direction, with only echos of the primary data returning from the other end (the hardware is the same as in full duplex).

A simplex channel is one in which information travels in just one direction. Typical applications might include: receive only printers, plotters, etc. Simplex channels are typically very easy to setup and develop software for.

A half-duplex channel is one in which information travels in just one direction at any given time, with the possibility of "turning the line around" (for informaiton to travel in the opposite direction). There is usually a relatively high overhead time associated with turning a line around. This type channel is typically found is older, less sophisticated equipment. Higher speed modems often employ half-duplex in order to obtain maximum utilization of a given bandwidth.

A full-duplex channel is one in which information may travel in both directions simultaneously. Note that such a channel may be used to emulate either of the simpler modes. This type channel typically is the most demanding on the communication hardware and software. One usually must resort to assembly language in order to provide the sort of concurrency required to support a full-duplex link.

Data Communications Hardware

There are several hardware "building blocks" used to create a communications system. Among these are computer "ports", terminals, multiplexors, line drivers, modems, and phone system interfaces.

Almost all computers have facilities for connection to one or more serial devices. These are typically implemented as "serial ports". Such a port may be as simple as a single UART with RS-232 conversion circuits and some mechanism to allow the CPU to read and write parallel data and status from (to) the UART. Such ports are typically implemented as DTE (Data Terminal Equipment) so that they may be connected to a DCE (Data Communication Equipment, such as a modem) with a simple (un-flipped) RS-232 cable. To connect such a port directly to another DTE (such as a terminal), a special cable is required which "flips" several signal pairs (note that the DTE and DCE ends of an RS-232 interface are symmetric). Some computers actually have smaller processors ("front ends") which oversee the operation of some (or

all) of the ports connected to the overall computer system. These front end processors typically free up the main CPU for more complex tasks, and "interrupt" it only when a mass transfer of data between the main CPU and front end is required. In a reasonable front end system, the main CPU would be interrupted only once an entire line had been received from a given terminal, and the front end would be able to accept an entire line of output at once for a given terminal. This means that the front end processor must have at least enough memory for both input and output buffers for each terminal connected through it.

There are a wide variety of currently available terminals which may be used in communications systems. These may be asynchronous or synchronous, hard copy or CRT, intelligent or "dumb".

By now, the distinction between asynchronous and synchronous should be clear. Suffice it to say that asynchronous terminals typically are character oriented, whereas synchronous terminals typically are message (line or page) oriented. This refers to the amount of information that is transmitted or received in a single operation. Most synchronous terminals are more "intelligent" than asynchronous terminals, so that a line (or page) of text may be entered and edited locally, and also so that a line protocol may be supported.

Hard copy terminals use some mechanism for physically imprinting (impact, thermal or otherwise) characters on paper. A CRT terminal employs a Cathode Ray Tube much like a TV picture tube to display characters and/or graphics. An RO terminal is Receive Only (no keyboard). A KSR terminal can both send (from the keyboard) and receive. An ASR (Automatic Send/Receive) terminal can also send and receive from some machine readable media, typically paper tape, cassette tape, or diskette.

An intelligent terminal typically has various "local" capabilities, such as forms generation, screen editing, possibly a calculator mode and/or standalone operation as a general purpose computer.

At times, it is necessary to "collapse" multiple low speed lines into a single high speed line, and/or expand such a line back into multiple low speed lines. This is usually done to save on phone line costs, or to take advantage of a wide bandwidth channel, such as coax cable. There are two mechanisms for "multiplexing" lines in this manner: time division and frequency division.

In a Time Division Multiplexor (TDM), the high speed line is "time sliced", typically at the character level. Certain recent such devices have used a technique called "statistical multiplexing", whereby more than "n" terminals running at baud rate "m" can be multiplexed onto a single line running at baud rate "nm", so long as not all of them run at full speed all the time. Note that this is a valid assumption most of the time. Note

that this is an ideal application for a dedicated microprocessor system.

In a Frequency Division Multiplexor (FDM), the bandwidth of an analog channel is divided into multiple sub-bands, and a seperate channel is maintained in each sub-band through simple frequency shifting techniques. The sub-bands are split back into seperate signals at the other end using bandpass filters and frequency down-shifters. This is similar to the method whereby the phone company multiplexes many 3 kilohertz wide voice channels onto a single physical wire.

A concentrator is a device which allows the first "n" phone lines out of "m" possible lines (n<=m) to actually connect through to modems, and hence to a computer system. The n+1st dial in gets a busy signal. This sort of device is quite useful at any site with more users than dial in ports (such as a university).

Modems are used to transduce serial digital data to/from analog audio frequency tones which may then be sent over voice-grade phone lines without undue distortion and/or disruption of other phone company services. Ma Bell herself markets an extensive (and expensive) line of modems which basically have set the standards for the industry as a whole. Most available modems claim to be compatible with one or more of Bell's standard models, hence it is sufficient to describe the characteristics of the Bell units.

Several different modulation techniques are used in modems, such as FSK (Frequency Shift Keying) and PM (Phase Modulation). FSK involves shifting a base (carrier) frequency up or down a given incremental frequency to represent a logic "1" or "0" respectively. PM is typically used in conjunction with FSK to achieve "multiple bit per baud" speeds. Most 4800 and 9600 bit per second modems actually run at 1200 baud, with either 4PM (4 phase) or 8PM (8 phase) modulation superimposed. Note that PM type modems tend to be quite expensive.

As modems are useful only in groups of two or more, and are often "symmetric" in nature, the terms "originate" and "answer" are used to distinguish between the two possible roles. Some full duplex modems are "originate only", some are "answer only", and some can swing either way. This primarily refers to which frequency they "listen" to, and which they "talk" to.

An originate modem typically transmits on the lower carrier frequency, and receives on the higher one. An answer modem is the mirror image of this (receives on the lower and transmits on the higher). Hence with full duplex modems, you need one (operating as an) Originate modem, and one (operating as an) Answer modem to establish a connection. The Answer modem is typically at a central computer site, and the Originate modem is typically at a remote terminal site.

With half-duplex modems (on a single phone circuit), a given

modem typically runs as an Originate modem for a while, then the line is "turned around" and it runs as an Answer modem for a while. Note that since data will only ever be travelling in one direction at any given time, only one carrier frequency is used.

Examples of Currently Available Modems

The Bell 103A is an asynchronous, originate/answer modem which has a maximum throughput of 300 bits per second. It uses simple FSK and can be run full duplex. Leases for $25-$35 per month. Most all modems used by hobbyists are compatible with this unit, such as the Novation Cat, D.C. Hayes Micromodem 100 and Apple modem, and so on.

The Bell 201C is a synchronous modem which uses 4PM. It has a maximum throughput of 2400 bits per second. With a single phone circuit (2 wire) it must be run half duplex. With a dual circuit (4 wire), it can be run full duplex. Leases for $70 per month.

The Bell 202C is an asynchronous modem which uses FSK. It has a maximum throughput of 1200 bits per second, and may be run only half duplex. The 202D allows full duplex operation at speeds up to 1800 bits per second over 4 wire circuits. Leases for $35-$50 per month.

The Bell 208B is a synchronous modem which uses 8PM. It has a maximum throughput of 4800 bits per second, and runs only half duplex. Leases for $150 per month.

Vadic has a unit which is not compatible with any Bell model, yet which is in widespread use (the VA3400 series). It is an aynchronous FSK modem with a maximum throughput of 1200 bits per second, and can run full duplex on a 2 wire circuit. Leases for $55 per month. This is used in many commercial timesharing systems, such as TYMNET and TELENET. Note that this device appears to the computer or terminal as a 103A type device, except for baud rate.

Bell has recently announced a 212A model, which can run full duplex 1200 baud asynchronous over a 2 wire circuit (as well as 103A style) to do battle with the Vadic 3400 (ah competition!).

Connection to the Switched Telephone Network

In order to protect your equipment from possible voltage spikes, etc. from the phone system, and vica versa, a DAA (Data Access Arrangement) is inserted in the line between a modem and the actual wall jack. This is simply a circuit which will not allow any kind of damaging signals to get through in either direction. The phone company used to lease these units for $5 per month or so, but no longer provides this service. DAA's are available for purchase from a number of companies (such as UDS). Most new modems being sold now include the DAA as an integral part of the modem (usually referred to as 'Direct Connect').

Some modem/DAA combinations are capable of detecting an incoming call, and either answering by themselves or alerting the device they are connected to, which can then request it to go 'off-hook'. Likewise, it is possible for a modem/DAA combination to originate a call (Auto-dial). This is sometimes done with touch tones, but usually with carefully timed clicks (on-hook / off-hook). If the device connected to the modem/DAA has a mechanism for going off-hook in order to answer an incoming call, it can also dial using timing loops. There is a Bell device called the 801C which can be used to implement Auto-dialing.

Some modems avoid the need for a DAA by using an "acoustic coupler". This is a speaker / microphone pair which allows the audio signal to be coupled into the phone system through the microphone / speaker in a standard handset. This makes it easier to connect into the phone system, but has the undesirable side effect of being much lower level (by a factor of 8 or so) than a direct connect system, hence is much more subject to error. The reason for this lower level is to prevent interference of the outgoing signal with the incoming one. A standard handset feeds some of the signal from the mouthpiece microphone back into the earpiece speaker to make the user feel more comfortable. This unfortunately does not make the acoutic coupler at all happy, so the transmit level must be held very low to prevent interference.

A normal, randomly chosen switched line on the phone system typically has a 3 kilohertz bandwidth, and widely separated "burst" noise characteristics, typically 10 msec in length, with an average error rate of one bit in 100,000. Better error rates can be obtained by going to "leased lines" with or without "conditioning" (which is very costly).

Simple Communications Systems

The first example of a communications system is a simple DTE-DCE link, such as a terminal to modem connection. In this case, we will need a straight (un-flipped) 8 line cable, on the following pins:

2	TD	Transmitted Data	DTE	-->	DCE
3	RD	Received Data	DTE	<--	DCE
4	RTS	Request To Send	DTE	-->	DCE
5	CTS	Clear To Send	DTE	<--	DCE
6	DSR	Data Set Ready	DTE	<--	DCE
7	SG	Signal Ground	DTE	---	DCE
8	DCD	Data Carrier Detect	DTE	<--	DCE
20	DTR	Data Terminal Ready	DTE	-->	DCE

If fewer lines are desired in the connecting cable, the absolute minimum is lines 2,3 and 7 with "wraparounds" on both ends to convince both parties that the other is really there. Note that in normal use, the terminal brings up DTR and the modem brings up DSR when they are powered on. Many terminals require an incoming DSR (and modems an incoming DTR) before they will operate. In an asynchronous, full duplex environment, the terminal then brings

up RTS, and awaits CTS from the modem (which is brought up, along with DCD, by the modem, once the incoming carrier has been detected). At this point, the terminal may transmit data to the modem over line 2, and the modem may transmit data to the terminal over line 3. In a half duplex system, the RTS and CTS lines are used to determine which direction the data will be going over the phone line.

If the terminal (DTE) were to be connected to another DTE (for example a serial printer), the following pairs would have to be flipped:

(2,3) (4,5) (6,20)

everything else should work as before.

As a slightly more elaborate communications system, we will connect a remote terminal to a local computer via the phone system:

The first interface is a simple RS232 DTE to DCE connection between the terminal and the originate modem (Bell 103A), as described in the previous example.

The second interface is between the modem and the phone system. In this case, we will be using an acoustic coupler, into a standard phone handset, using manual dialing.

The third interface is between the phone system and the answer modem. In this case, we will use a direct connect auto answer modem, such as the D-CAT (also Bell 103A).

The fourth interface is another simple DCE to DTE link from the answer modem to a computer port which is configured as a DTE, as described in the previous example.

Once software is installed to handle handshaking and auto answer, this communications system allows the remote terminal to dial in and use the system as if connected directly to it at 300 baud.

Appendix B

A Practical Guide to RS-232 Interfacing

The following information is intended to collect together in one place, and explain in relatively simple terms, enough of the details of the RS-232 standard to allow a technician to construct and/or debug interfaces between any two "RS-232 Compatible" devices. A more detailed coverage of the subject may be found in the book "Technical Aspects of Data Communication" by John E. McNamara (1977, Digital Press).

This guide is necessary due to the casual way that vendors implement "RS-232" interfaces, sometimes omitting required signals, requiring optional ones, or worse, implementing signals incorrectly. Due to this, and a lack of readily available information about the real EIA standard, there is often considerable confusion involved in trying to interface two RS-232 devices.

BACKGROUND

RS-232-C is the most recent version of the EIA (Electronics Industry Association) standard for low speed serial data communication. It defines a number of parameters concerning voltage levels, loading characteristics and timing relationships. The actual connectors which are almost universally used (DB-25P and DB-25S, sometimes called "EIA connectors") are recommended, but not mandatory. Typical practice requires mounting the female (DB-25S) connector on the chassis of communication equipment, and male (DB-25P) connectors on the cable connecting two such devices.

There are two main classes of RS-232 devices, namely DTE (Data Terminal Equipment), such as terminals, and DCE (Data Communication Equipment), such as modems. Typically, one only interfaces a DTE to a DCE, as opposed to one DTE to another DTE, or one DCE to another DCE, although there are ways to do the later two by building non-standard cables. Rarely if ever are more than two devices involved in a given interface (multidrop is not supported). A serial port on a computer may be implemented as either DTE or DCE, depending on what type of device it is intended to support.

RS-232 is intended for relatively short (50 feet or less), relatively low speed (19,200 bits per second or less) serial (as opposed to parallel) communications. Both asynchronous and synchronous serial encoding are supported. As 'digital' signals (switched D.C. voltage, such as square waves) are used, as opposed to 'analog' signals (continuously varying voltage, such as sine waves) a very wide bandwidth channel (such as direct wire) is required. A limited bandwidth channel (such as a phone circuit) would cause severe and unacceptable distortion and consequent loss of information.

RS-232 will support simplex, half-duplex, or full-duplex type channels. In a simplex channel, data will only ever be travelling in one direction, e.g. from DCE to DTE. An example might be a 'Receive Only' printer. In a half-duplex channel, data may travel in either direction, but at any given time data will only be travelling in one direction, and the line must be 'turned around' before data can travel in the other direction. An example might be a Bell 201 style modem. In a full-duplex channel, data may travel in both directions simultaneously. An example might be a Bell 103 style modem. Certain of the RS-232 'hand-shaking' lines are used to resolve problems associated with these modes, such as which direction data may travel at any given instant.

If one of the devices involved in an RS-232 interface is a real modem (especially a half-duplex modem), the 'hand-shaking' lines must be supported, and the timing relationships between them are quite important. These lines are typically much easier to deal with if no modems are involved. In certain cases, these lines may be used to allow one device (which is receiving data at a higher rate than it is capable of processing indefinitely) to cause the other device to pause while the first one 'catches up'. This use of the hand-shaking lines was not really intended by the designers of the RS-232 standard, but it is a useful by-product of the way such interfaces are typically implemented.

Much of the RS-232 standard is concerned with support of 'modems'. These are devices which can convert a serial digital data signal into an analog signal compatible with a narrow bandwidth (e.g. 3 kHz) channel such as a switched telephone circuit, and back into serial digital data on the other end. The first process is called 'MOdulation', and the second process is called 'DEModulation', hence the term 'MODEM'. The actual process used (at data rates of up to 1200 bits per second) is FSK (Frequency Shift Keying), in which a constant frequency sine wave (called the 'carrier') is shifted to a slightly higher or slightly lower frequency to represent a logic 0 or logic 1, respectively. In a half duplex modem, the entire available bandwidth is used for one direction. In a full duplex modem, the available bandwidth is divided into two sub-bands, hence there is both an 'originate carrier' (e.g. for data from the terminal to the computer), and an 'answer carrier' (e.g. for data from the computer to the terminal). The actual frequencies (in Hertz) used on the Bell 103A full duplex modem are:

signal	state	Originate	Answer
logic 0	SPACE	1180	1850
carrier		1080	1750
logic 1	MARK	980	1650

THE STANDARD CIRCUITS AND THEIR DEFINITIONS

For the purposes of the RS-232 standard, a 'circuit' is defined to be a continuous wire from one device to the other.

There are 25 circuits in the full specification, less than half of which are at all likely to be found in a given interface. In the simplest case, a full-duplex interface may be implemented with as few as 3 circuits. There is a certain amount of confusion associated with the names of these circuits, partly because there are three different naming conventions (common name, EIA circuit name, and CCITT circuit name). The table below lists all three names, along with the circuit number (which is also the connector pin with which that circuit is normally associated on both ends). Note that the signal names are from the viewpoint of the DTE (e.g. Transmit Data is data being sent by the DTE, but received by the DCE).

PIN	NAME	EIA	CCITT	DTE DCE	FUNCTION
1	CG	AA	101	---	Chassis Ground
2	TD	BA	103	-->	Transmit Data
3	RD	BB	104	<--	Receive Data
4	RTS	CA	105	-->	Request To Send
5	CTS	CB	106	<--	Clear To Send
6	DSR	CC	107	<--	Data Set Ready
7	SG	AB	102	---	Signal Ground
8	DCD	CF	109	<--	Data Carrier Detect
9*				<--	Pos. Test Voltage
10*				<--	Neg. Test Voltage
11					(usually not used)
12+	SCDC	SCF	122	<--	Sec. Data Car. Detect
13+	SCTS	SCB	121	<--	Sec. Clear To Send
14+	STD	SBA	118	-->	Sec. Transmit Data
15#	TC	DB	114	<--	Transmit Clock
16+	SRD	SBB	119	<--	Sec. Receive Data
17#	RC	DD	115	<--	Receive Clock
18					(not usally used)
19+	SRTS	SCA	120	-->	Sec. Request To Send
20	DTR	CD	108.2	-->	Data Terminal Ready
21*	SQ	CG	110	<--	Signal Quality
22	RI	CE	125	<--	Ring Indicator
23*		CH	111	-->	Data Rate Selector
		CI	112	<--	Data Rate Selector
24*	XTC	DA	113	-->	Ext. Transmit Clock
25*				-->	Busy

In the above, the character following the pin number means:

* rarely used
\+ used only if secondary channel implemented
\# used only on synchronous interfaces

also, the direction of the arrow indicates which end (DTE or DCE) originates each signal, except for the ground lines (---). For example, circuit 2 (TD) is originated by the DTE, and received by the DCE. Certain of the above circuits (11, 14, 16, and 18) are used only by (or in a different way by) Bell 208A modems.

A secondary channel is sometimes used to provide a very slow

(5 to 10 bits per second) path for return information (such as ACK or NAK characters) on a primarily half duplex channel. If the modem used suppports this feature, it is possible for the receiver to accept or reject a message without having to 'turn the line around', a process that usally takes 100 to 200 milliseconds.

On the above circuits, all voltages are with respect to the Signal Ground (SG) line. The following conventions are used:

Voltage	Signal	Logic	Control
+3 to +25	SPACE	0	On
-3 to -25	MARK	1	Off

Note that the voltage values are inverted from the logic values (e.g. the more positive logic value corresponds to the more negative voltage). Note also that a logic 0 corresponds to the signal name being 'true' (e.g. if the DTR line is at logic 0, that is, in the +3 to +25 voltage range, then the Data Terminal IS Ready).

ELECTRICAL CHARACTERISTICS OF EACH CIRCUIT

The following criteria apply to the electrical characteristics of each of the above lines:

1) The magnitude of an open circuit voltage shall not exceed 25V.

2) The driver shall be able to sustain a short to any other wire in the cable without damage to itself or to the other equipment, and the short circuit current shall not exceed 0.5 ampere.

3) Signals shall be considered in the MARK (logic 1) state when the voltage is more negative than -3V with respect to the Signal Ground. Signals shall be considered in the SPACE (logic 0) state when the voltage is more positive that 3V with respect to the Signal Ground. The range between -3V and 3V is defined as the transition region, within which the signal state is not defined.

4) The load impedance shall have a DC resistance of less than 7000 ohms when measured with an applied voltage of from 3V to 25V but more than 3000 ohms when measured with a voltage of less than 25V.

5) When the terminator load resistance meets the requirements of Rule 4 above, and the terminator open circuit voltage is 0V, the magnitude of the potential of that circuit with respect to Signal Ground will be in the 5V to 15V range.

6) The driver shall assert a voltage between -5V and -15V relative to the signal ground to represent a MARK signal condition. The driver shall assert a voltage between 5V and 15V relative to the Signal Ground to represent a SPACE signal condition. Note that this rule in conjunction with Rule 3 above allows for 2V of noise margin. Note also that in practice, -12V

and 12V are typically used.

7) The driver shall change the output voltage at a rate not exceeding 30 volts per microsecond, but the time required for the signal to pass through the -3V to +3V transition region shall not exceed 1 millisecond, or 4 percent of a bit time, whichever is smaller.

8) The shunt capacitance of the terminator shall not exceed 2500 picofarads, including the capacitance of the cable. Note that when using standard cable with 40 to 50 picofarads per foot capacitance, this limits the cable length to no more than 50 feet. Lower capacitance cable allows longer runs.

9) The impedance of the driver circuit under power-off conditions shall be greater than 300 ohms.

Note that two widely available integrated circuit chips (1488 and 1489) implement TTL to RS232 drivers (4 per chip), and RS232 receivers to TTL (also 4 per chip), in a manner consistent with all of the above rules.

DEFINITION OF THE MOST COMMON CIRCUITS

1 CG Chassis Ground

This circuit (also called Frame Ground) is a mechanism to insure that the chassis of the two devices are at the same potential, to prevent electrical shock to the operator. Note that this circuit is not used as the reference for any of the other voltages. This circuit is optional. If it is used, care should be taken to not set up ground loops.

2 TD Transmit Data

This circuit is the path whereby serial data is sent from the DTE to the DCE. This circuit must be present if data is to travel in that direction at any time.

3 RD Receive Data

This circuit is the path whereby serial data is sent from the DCE to the DTE. This circuit must be present if data is to travel in that direction at any time.

4 RTS Request To Send

This circuit is the signal that indicates that the DTE wishes to send data to the DCE (note that no such line is available for the opposite direction, hence the DTE must always be ready to accept data). In normal operation, the RTS line will be OFF (logic 1 / MARK). Once the DTE has data to send, and has determined that the channel is not busy, it will set RTS to ON (logic 0 / SPACE), and await an ON condition on CTS from the DCE, at which time it may then

begin sending. Once the DTE is through sending, it will reset RTS to OFF (logic 1 / MARK). On a full-duplex or simplex channel, this signal may be set to ON once at initialization and left in that state. Note that some DCEs must have an incoming RTS in order to transmit (although this is not strictly according to the standard). In this case, this signal must either be brought across from the DTE, or provided by a wraparound (e.g. from DSR) locally at the DCE end of the cable.

5 CTS Clear To Send

This circuit is the signal that indicates that the DCE is ready to accept data from the DTE. In normal operation, the CTS line will be in the OFF state. When the DTE asserts RTS, the DCE will do whatever is necessary to allow data to be sent (e.g. a modem would raise carrier, and wait until it stabilized). At this time, the DCE would set CTS to the ON state, which would then allow the DTE to send data. When the RTS from the DTE returns to the OFF state, the DCE releases the channel (e.g. a modem would drop carrier), and then set CTS back to the OFF state. Note that a typical DTE must have an incoming CTS before it can transmit. This signal must either be brought over from the DCE, or provided by a wraparound (e.g. from DTR) locally at the DTE end of the cable.

6 DSR Data Set Ready

This circuit is the signal that informs the DTE that the DCE is alive and well. It is normally set to the ON state by the DCE upon power-up and left there. Note that a typical DTE must have an incoming DSR in order to function normally. This line must either be brought over from the DCE, or provided by a wraparound (e.g. from DTR) locally at the DTE end of the cable. On the DCE end of the interface, this signal is almost always present, and may be wrapped back around (to DTR and/or RTS) to satisfy required signals whose normal function is not required.

7 SG Signal Ground

This circuit is the ground to which all other voltages are relative. It must be present in any RS-232 interface.

8 DCD Data Carrier Detect

This circuit is the signal whereby the DCE informs the DTE that it has an incoming carrier. It may be used by the DTE to determine if the channel is idle, so that the DTE can request it with RTS. Note that some DTEs must have an incoming DCD before they will operate. In this case, this signal must either be brought over from the DCE, or provided locally by a wraparound (e.g. from DTR) locally at the DTE end of the cable.

15 TC Transmit Clock

This circuit provides the clock for the transmitter section of a synchronous DTE. It may or may not be running at the same rate as the receiver clock. This circuit must be present on synchronous interfaces.

17. RC Receiver Clock

This circuit provides the clock for the receiver section of a synchronous DTE. It may of may not be running at the same rate as the transmitter clock. Note that both TC and RC are sourced by the DCE. This circuit must be present on synchronous interfaces.

20 DTR Data Terminal Ready

This circuit provides the signal that informs the DCE that the DTE is alive and well. It is normally set to the ON state by the DTE at power-up and left there. Note that a typical DCE must have an incoming DTR before it will function normally. This signal must either be brought over from the DTE, or provided by a wraparound (e.g. from DSR) locally at the DCE end of the cable. On the DTE side of the interface, this signal is almost always present, and may be wrapped back around to other circuits (e.g. DSR, CTS and/or DCD) to satisfy required hand-shaking signals if their normal function is not required.

Note that in an asynchronous channel, both ends provide their own internal timing, which (as long as they are within 5% of each other) is sufficient for them to agree when the bits occur within a single character. In this case, no timing information need be sent over the interface between the two devices. In a synchronous channel, however, both ends must agree when the bits occur over possibly thousands of characters. In this case, both devices must use the same clocks. Note that the transmitter and receiver may be running at different rates. Note also that BOTH clocks are provided by the DCE. When one has a synchronous terminal tied into a synchronous port on a computer via two synchronous modems, for example, and the terminal is transmitting, the terminal's modem supplies the Transmit Clock, which is brought directly out to the terminal at its end, and encodes the clock with the data, sends it to the computer's modem, which recovers the clock and brings it out as the Receive Clock to the computer. When the computer is transmitting, the same thing happens in the other direction. Hence, whichever modem is transmitting must supply the clock for that direction, but on each end, the DCE device supplies both clocks to the DTE device.

All of the above applies to interfacing a DTE device to a DCE device. In order to interface two DTE devices, it is usually sufficient to provide a 'flipped' cable, in which the pairs (TD, RD), (RTS,CTS) and (DTR,DSR) have been flipped. Hence, the TD of

one DTE is connected to the RD of the other DTE, and vica versa. It may be necessary to wrap various of the hand-shaking lines back around from the DTR on each end in order to have both ends work. In a similar manner, two DCE devices can be interfaced to each other.

An RS-232 'break-out box' is particularly useful in solving interfacing problems. This is a device which is inserted between the DTE and DCE. Firstly, it allows you to monitor the state of the various hand-shaking lines (light on = signal ON / logic 0), and watch the serial data flicker on TD and/or RD. Secondly, it allows you to break the connection on one or more of the lines (with dip-switches), and make any kind of cross-connections and/or wraparounds (with jumper wires). Using this, it is fairly easy to determine which line(s) are not functioning as required, and quickly build a prototype of a cable that will serve to interface the two devices. At this point, the break-out box can be removed and a real cable built that performs the same function. An example of this kind of device is the International Data Sciences, Inc. Model 60 'Modem and Terminal Interface Pocket Analyzer' (also called a 'bluebox'). Care should be taken with this type of device to connect the correct end of it to the DTE device, or the lights and switches do not correspond to the actual signals.

Appendix C

The Text Compression / Expansion Utilities - Dick Greenlaw

Note: The following documentation, and the software described therein is included free of charge (with permission) to save the user from having to download these programs from an RCPM system. They are not supported by Mycroft Labs, but are highly recommended, and appear to work quite well, as documented. They are required for normal use of most RCPM systems, as many text files on these systems are stored in squeezed form, to keep disk storage space and transmission time to a minimum. You may even find them useful in exchanging files with other users. Feel free to share these programs with friends as covered below.

USAGE DOCUMENTATION FOR: 7/18/81

SQ.COM	1.3	File squeezer
USQ.COM	1.4	File unsqueezer
FLS.COM	1.1	Ambiguous file name expander

DISTRIBUTION RIGHTS:

I allow unrestricted non-profit distribution of this software and invite users groups to spread it around. However, any distribution for profit requires my permission in advance. This applies only to the above listed programs and their program source and documentation files. I do sell other software.

PURPOSE:

The file squeezer, SQ, compresses files into a more compact form. This provides:

1. Faster transmission by modem.

2. Fewer diskettes to distribute a program package. (Include USQ.COM and instructions, both unsqueezed.)

3. Fewer diskettes for archival storage.

Any file can be squeezed, but program source files and text files benefit the most, typically shrinking by 35%. Files containing only a limited character set, such as dictionary files, may shrink as much as 48%. Squeezed files look like gibbersh and must be unsqueezed before they can be used.

The unsqueezer, USQ, expands squeezed files into exact duplicates of the original or provides a quick, unsqueezed display of the tops of (or all of) squeezed files. Unsqueezing requires only a single pass.

Both SQ and USQ accept batches of work specified by lists of file names (with drives if needed) and miscellaneous options. They accept these parameters in any of three ways:

1. On the CP/M command line.
2. From the console keyboard.
3. From a file.

The FLS program can be used (on the same command line!) to expand parameter lists containing wild-card (ambiguous) file names into lists with the specific file names required by SQ and USQ.

This combination of programs allows you to issue a single command which will produce many squeezed or unsqueezed files from and to various diskettes. For example, to unsqueeze all squeezed ASM files on drive B and send the results to drive C and also unsqueeze all squeezed TXT files on drive A and send the results to drive D:

```
A>fls c: b:*.aqm d: *.tqt |usq
```

For detailed instructions see USAGE. This DOES run under plain old vanilla CP/M! Many of the smarts are buried in the COM files in the form of library routines provided with the BDS C package (available from Lifeboat).

The above example simulates a "pipe" (indicated by the "|") by sending the "console" output of the fls.com program to a temporary file and then running the sq.com program with options which cause it to read its parameters from its "console" input, which is really redirected to come from the temporary file.

THEORY:

The data in the file is treated at the byte level rather then the word level, and can contain absolutely anything. The compression is in two stages: first repeated byte values are compressed and then a Huffman code is dynamically generated to match the properties of each particular file. This requires two passes over the source data.

The decoding table is included in the squeezed file, so squeezing short files can actually lengthen them. Fixed decoding tables are not used because English and various computer languages vary greatly as to upper and lower case proportions and use of special characters. Much of the savings comes from not assigning codes to unused byte values.

More detailed comments are included in the source files.

USAGE TUTORIAL:

As usual, you have to learn how to tell the programs what to do (i.e., what parameters to type after the program name). First I will introduce the various possibilities by example. Then I will summarize the rules.

In the simplest case either SQ or USQ can simply be given one or more file names (with or without drive names):

```
A>sq xyz.asm

A>sq thisfile.doc b:thatfile.doc
```

will create squeezed files xyz.aqm, thisfile.dqc and thatfile.dqc, all on the current drive, A. The original files are not disturbed. Note that the names of the squeezed files are generated by rules - you don't specify them.

Likewise,

```
A>usq xyz.aqm
```

will create file xyz.asm on the A drive, overwriting the original. (The original name is recreated from information stored in the squeezed version.) The squeezed version is not disturbed.

Each file name is processed in order, and you can list all the files you can fit in a command. The file names given to SQ and USQ must be specific. You will learn below how to use the FLS program to expand patterns like *.asm (all files of type asm) into a list of specific names and feed them into SQ or USQ.

The above examples let the destination drive default to the current logged drive, which was shown in the prompt to be A. You can change the destination drive as often as you like in the parameter list. For example,

```
A>sq x.asm b: y.asm z.asm c: d:s.asm
```

will create x.aqm on the current drive, A, y.aqm and z.aqm on the B drive and s.aqm on the C drive. Note that the first three riginals are on drive A and the last one is on drive D. Remember that each parameter is processed in order, so you must change the destination drive before you specify the files to be created on that drive.

Eventually you will have diskettes with many squeezed files on them and you will wonder what is in which file. If they weren't squeezed you would use the TYPE command to look at the comments at the beginning of the files. But squeezed files just make a mess on your CRT screen when you TYPE them, so I have provided the required feature as a preview option to the USQ program.

```
A>usq -10 x.bas b:y.asm
```

will not take the time to create unsqueezed files. Instead it will unsqueeze the first 10 lines of each file and display them on your console. The display from each file consists of the file names, the data and a formfeed (FF). Also,

```
A>usq - c:xyz.mac
```

will unsqueeze and display the first 65,535 lines of any files listed. That's the biggest number you can give it, and is intended to display the whole file.

This preview option also ensures that the data is displayable. The parity bit is stripped off (some Wordstar files use it for format control) and any unusual control characters are converted to periods. You'll see some of these at the end of the files as the CP/M end of file is treated as data and the remainder of the sector is displayed.

You are now familiar with all of the operational parameters of SQ and USQ. But so far you have always typed them on the command line which caused the program to be run. For reasons which will become apparent later, I have also provided an interactive mode. If there are no parameters (except directed i/o parameters, described later) on the command line, SQ and USQ will prompt with an asterisk and accept parameters from the console keyboard. Each parameter must be followed by RETURN and will be processed immediately. An empty command (just RETURN) will cause the program to exit back to CP/M. Try it - it will help you understand what follows.

Now lets get into directed i/o, which will be new to most of you, but will save you so much work you will wonder how you ever got along without it.

Perhaps you frequently squeeze or unsqueeze the same list of files and you would like to type the list once and be done with it. Use an editor (or FLS, described below) to create a file with one parameter per line. For example call it commands.lst.

Then,

```
A>sq <commands.lst
```

will cause the command list file to be read as if you were typing it! You will see it on the console.

That was redirected console input. Now assume that you have a very long list of files to squeeze or unsqueeze and while you are taking a nap the progress comments and maybe some error comments scroll off the screen. Redirecting the console output will let you capture the progress information in a file so you can check it later. The error comments will have the screen to themselves.

For example,

```
A>sq <commands.lst >out
```

will send the progress comments to the file "out", which you can TYPE later. The routine display of the program name and version, etc., will still go to the console.

A more practical example is to send that information to the console and to the file.

```
A>sq <commands.lst +out
```

will do that.

Redirected input and output are independent - you can do either, both or neither.

There is one more form of redirection called a "pipe". It is by far the most important to you. Recall that I promised to tell you how to use ambiguous file names such as *.asm (all files of type asm on the current default drive) or *.?q? (all files having a "q" as the second letter of their type). That last example just happens to mean "all squeezed files", assuming you don't have any other files with such a silly name (I hope).

I have provided a program called FLS which is intended primarily for use in pipes. Here is an example:

```
A>fls c: x.asm y*.asm >temp.$$$
```

will simply pass the first two parameters through to the console output, which is being redirected to a file called temp.$$$. But the third parameter will be replaced by all the files on the current drive which are of type asm and have names beginning with y.

FLS is smart enough to know that a letter followed by a colon and nothing else is a destination drive name intended for SQ or USQ. It will also treat any parameter beginning with a - (minus sign) as an option to be passed through. Anything else is considered a file name or pattern and is checked against the directory of the appropriate drive.

Therefore you could use:

```
A>fls b: c:*.aqm *.aqm -10 stuff.dqc >temp.$$$
A>usq <temp.$$$
A>era temp.$$$
```

to unsqueeze all files of type aqm on drives C and A and put the unsqueezed files on drive B, and then preview the first 10 lines of file stuff.dqc.

Here is where the pipe comes in. The above three commands can be abbreviated as:

```
A>fls b: c:*.aqm *.aqm -10 stuff.dqc |usq
```

That little "|" is the pipe option and it causes the FLS output to be redirected to a temporary file and when that is done it actually runs USQ for you with the proper input redirection and then erases the temporary file.

If that isn't enough, you can still use the + or > redirection option at the end of that line to capture the console output from USQ.

```
A>fls b: c:*.aqm *.aqm -10 stuff.dqc |usq >out
```

If you plan your comments carefully you can produce a single file containing an abstract of an entire library of squeezed files in one step!

```
A>fls -25 *.?q? |usq >abstract
```

One final point. Anywhere you specify a file name you can specify a drive in front of it. That applies to redirection and well as files to be squeezed and unsqueezed. If a name begins with a - (minus sign) it will look like an option to FLS unless you put a drive name in front of it (b:-sq.077).

USAGE SUMMARY:

The previous section gradually presented the various options by example. This section gives a condensed and more abstract description and is intended for reference. If you couldn't see the forest for the trees, maybe this will give you a better view.

The parameter handling of these programs is straightforward. Parameters fall into two classes: directed i/o options and operational parameters. Note that parameters read from files or from the console are not forced to upper case, but the internal file handling routines all treat lower case as upper case.

When a file to be written already exists, it is quietly overwritten.

Directed I/O parameters:

The first action taken by these programs is to process directed i/o parameters from the CP/M command line. These parameters are optional and take the forms:

<file	read console input from file
>file	send most console output to file
+file	send most console output to file and console
|pgm ...	send most console output to a temporary file then run PGM.COM and take console input from the temporary file. "..." represent the parameters for PGM. This is called "piping".

Only one input and one output redirection can apply to each program. After the program has arranged for any directed i/o parameters to be obeyed they are deleted from the parameter list

seen by the rest of the program.

Operational parameters:

The program then checks if there are any remaining parameters from the CP/M command line. If there are, they are obeyed. If and only if there are no remaining parameters on the command line, the program prompts for them at the console. If console input has been directed to a file one parameter is read and obeyed from each line of the file. Otherwise, the user follows each typed parameter with a RETURN and an empty command exits the program.

Each operational parameter is obeyed without looking ahead to other parameters, so options should precede the file names to which they apply.

SQ operational parameters are a list of the following types:

drive:	set the current destination drive
filename	file to be squeezed
drive:filename	" " " "

SQ does not change the files being squeezed. New, squeezed files are created on the destination drive (defaults to the current drive) with names derived from the original name but with the second letter of the file type (extention) changed to Q. When there is no type, QQQ is used. The original name is saved in the squeezed file.

USQ operational parameters are a list of the following types:

drive:	set the current destination drive
filename	file to be squeezed
drive:filename	" " " "
-count	Preview (display on the console) the first "count" lines of each file, where "count" is a number from 1 to 65535.

If the -count option IS NOT in effect then USQ creates unsqueezed versions of the listed files on the destination drive, which defaults to the current logged drive. Each unsqueezed file is CRC checked against the CRC value of the original file, which is part of the squeezed file.

The -count option is for previewing squeezed files. It allows you to skim through a group of squeezed files, peeking at the first "count" lines in each. The > or + output redirection option could be used to capture this information in a file, along with the corresponding file names, thus forming an abstract of the files on a disk.

When the -count option is used the CRC check is cancelled and the output is forced into printable form by stripping the parity bit and changing most unprintable characters to periods. The exceptions are CR, LF, TAB and FF. The output from each file is terminated by an FF. PIP can be used to strip FFs and provide formatted printing if desired. "Count" defaults to the maximum value, 65535, in case you want to look at a whole file.

FLS operational parameters: FLS is a "filter", which means it accepts input from the console input or command line and transforms the input according to a set of rules to produce console output. That's fine for getting familiar with FLS, but to make it useful you "pipe" its output to the input of SQ or USQ.

Any FLS parameter which is of the form:

drive:

or -anything

is copied to console output unchanged.

Any other FLS operational parameter is treated as a file name and is checked against the directory of the appropriate drive. If it contains * or ? it is replaced by a list of all the files which fit the pattern. If nothing is found in the directory an error comment is sent to the console, even if normal console output has been redirected to a file.

IMPORTANT: when using a pipe from FLS or any other input redirection to get the file list, etc., on which USQ or SQ are to operate you must NOT put any parameters other than redirection following the program name. They must be all together in the input parameter list. Example:

```
A>fls -10  b:*.cq |usq +saveout
```

is the proper way to preview the top (first 10 lines) of each squeezed .C file on the B drive. The -10 is passed through FLS to USQ. The results will be displayed on the console and saved in file "saveout" on the A drive. The saveout file lets you confirm the list of processed files even if the display scrolls off the screen while running unattended.

In summary, i/o redirection parameters (those prefixed by +, <, >, or |) always follow the command to which they apply, but operational parameters (destination drive, -options) must be with the file name list.

EXAMPLES:

1. Unsqueeze all squeezed files on the current drive and put the resulting unsqueezed files on the same drive.

```
A>fls *.?q? |usq
```

2. Look at the first 10 lines of every squeezed file on drive B.

```
A>fls -10 b:*.?Q? |usq
```

note that since the file names for USQ came from FLS, the count option had to come from there too.

4. Squeeze all .ASM files on the B and C drives and put the squeezed files on the D drive.

```
A>fls d: b:*.asm c:*.asm |sq
```

Note that if d: had not been first the squeezed files would have gone to the A drive.

5. Squeeze file xyz.c on the A drive and put the results on the A drive.

```
A>sq xyz.c
```

6. Build a parameter list of all ASM files on drive C in file XX.PAR and view it on the console.

```
A>fls c:*.asm +xx.par
```

7. Use the above list to squeeze the files to the A drive.

```
A>sq <xx.par
```

8. As above, but results to the B drive.

```
A>b:
B>a:sq <a:xx.par
```

9. Squeeze all ASM and C files on the A drive and put the results on the B drive. Capture the progress comments in the file "out" without displaying them.

```
A>fls b: *.asm *.c |sq >out
```

10. Preview the first 24 lines of each squeezed ASM file THEN unsqueeze them (unless stopped via cntl-C).

```
A>fls -24 *.aqm a: *.aqm |usq
```

Note that specification of a destination drive cancels previewing.

IN CASE OF TROUBLE:

I welcome suggestions and bug reports, but you must understand that some of the ideas I get would involve almost as much program

development as the original package. I have what I want and (I hope) what most users want, so I am not motivated to spend many more months creating something entirely different which just happens to involve data compression. The data compression routines are probably less than half of this package, and are designed to operate on large blocks of data, such as files.

Dick Greenlaw
251 Colony Ct.
Gahanna, Ohio 43230

614-475-0172 weekends and evenings

Appendix D

Mycroft Labs Utilities

TRSCPM - TRSDOS to CP/M Text File Conversion

TRSCPM is a simple utility for converting text files from TRSDOS format to CP/M format. It is included with MITE because there is a popular datacomm program available under TRSDOS, called MODEM-80 that is compatible with MITE (using the XMODEM protocol). However, TRSDOS uses a slightly different text file format, hence a conversion utility is necessary. MODEM-80 includes a means of converting CP/M text files to TRSDOS format.

The command syntax is as follows:

```
TRSCPM I=afn
```

where 'afn' is an ambiguous filename that matches all of the files that you want to convert (e.g. *.TXT). The name of each file being converted will be printed to the console as TRSCPM executes. The converted files are written back onto the original files.

COMHEX - Translate .COM file to .HEX object file format

COMHEX is a simple utility program to create a HEX file from a COM file, for purposes of shipping object code to other systems that may not support the CLINK SEND/RECV protocol. To use it, just type 'COMHEX ufn' where ufn is the name field of a particular .COM file on the current disk. COMHEX will read the file 'ufn.COM', and write the file 'ufn.HEX'. It essentially performs the inverse of the standard CP/M LOAD command.

MFT - Multiple File Transfer utility for single drive systems

MFT is a transient program which runs under the Digital Research CP/M operating system. It is used for transferring one or more arbitrarily large files from one diskette to another. It is run by typing its name followed by a string of up to 64 CP/M file names (no disk drive id's), separated by at least one blank. It will attempt to read all specified files into memory, up to the start of the BDOS. If any file extends past the available memory area, any files and/or part thereof which were read will be written to the output disk, and a further pass will be requested. MFT will continue read/write passes until all files have been transferred.

The following example illustrates how MFT works:

```
A>mft mft.asm mft.com tps.asm sid.com mac.com load.com

MFT 2.0
Buffer size =   402 sectors
Mount input disk, type CR
MFT      ASM -    86 sectors read
MFT      COM -    12 sectors read
TPS      ASM -   304 sectors read
Mount output disk, type CR
MFT      ASM -    86 sectors written
MFT      COM -    12 sectors written
TPS      ASM -   304 sectors written
Mount input disk, type CR
TPS      ASM -   359 sectors read
SID      COM -    43 sectors read
Mount output disk, type CR
TPS      ASM -   359 sectors written
SID      COM -    43 sectors written
Mount input disk, type CR
SID      COM -    13 sectors read
MAC      COM -    92 sectors read
LOAD     COM -    14 sectors read
Mount output disk, type CR
SID      COM -    13 sectors written
MAC      COM -    92 sectors written
LOAD     COM -    14 sectors written

A>
```

Note that three passes were required to transfer all files. Also note that TPS was partially transferred in pass one (304 of its 663 sectors) with the remainder (359 sectors) being transferred in pass two. Note that the name TPS.ASM occurs both as the last file of pass one and as the first file of pass two. The file SID.COM was split in a similar manner between passes two and three.

A number of self-explanatory error messages may print out during normal use, if read or write errors occur at any point,

for example if the disk fills up during a write. No checks are made to prevent overwiting an existing file on the output disk.

Note that in version 3.0, any filename may actually be an ambigous file name (afn) as found elsewhere in CP/M. The character '?' may be used as a wildcard which will match any character, and the character '*' may be used to fill out the rest of either field (fn or ft) with '?' wildcards. Hence the name 'f*.asm' is equivalent to 'f???????.asm'. An entire disk may be backed up with the command 'MFT *.*'. Note that MFT will list all files which statisfy any ambiguous file names, as it reads them into memory.

As MFT creates its filename table, it may issue diagnostics which indicate that a particular filename was not found, or a syntax error was detected. MFT will struggle bravely on and ignore any such attempts at levity. Just those files which are legal names and present on the disk will be listed as it reads them into memory.

E - Line Numbered Text Editor for CP/M

E is a simple line-numbered editor program, created by Mycroft Labs, similar to the standard line-numbered editors found in most BASIC interpreters, such as Microsoft MBASIC. Unlike most line editors, this one is suitable for entering arbitrary text files or source programs for any language, such as BASIC-E, CBASIC, FORTRAN, ASM or MAC. This is due to the fact that E will work with files which do not have line numbers on them either before or after editing. In fact one may use E for entering any text, such as this documentation.

Anyone who is familiar with the editing conventions used in a typical BASIC interpreter can start off creating files with E with very little training. Those who are willing to read a short explanation of the features will find that a few useful extensions beyond the run-of-the-mill line editor have been provided.

The conventions adopted in this editor are as follows:

1) Typing a line number followed by a line of text will cause that text to be entered into the memory buffer after the line whose line number is the greatest number less than the current line number. This means that lines will wind up in numerical sequence by line number.

2) If there is already a line in the text buffer with the number of a new line, the new line will replace the old one. No warning is issued when this occurs.

3) If a line number is typed by itself, any line in the text buffer with that same number will be deleted.

4) If a line starts with a non-numeric character (unless in Auto-line Mode), that line is interpreted as a command. If not recognized, an error message is issued.

Several commands are provided to make editing simpler:

LIST	List all lines in the buffer
LIST n	List line n
LIST n-	List all lines starting with line n
LIST n-m	List all lines n through m inclusive
LIST n-m /str/	List all lines as above that include /str/
NEW	Erase all lines in buffer
KILL	Same as NEW command
KILL n	Erase line n
KILL n-	Erase all lines starting with line n
KILL n-m	Erase all lines n through m inclusive
AUTO	Start Auto-Line mode, FIRST = 10, INCR = 10
AUTO n	Start Auto-line mode, FIRST = n, INCR = 10

AUTO n,m	Start Auto-Line mode, FIRST = n, INCR = m
RESEQ	Resequence all lines, FIRST = 10, INCR = 10
RESEQ n	Resequence all lines, FIRST = n, INCR = 10
RESEQ n,m	Resequence all lines, FIRST = n, INCR = m
CAPS	Set flag to translate all lower case to caps
SPACE	Request status of available space in buffer
QUIT	Abort without writing buffer to disk
END	Write buffer to disk, rename files

Once auto-line mode is initiated with the AUTO command, E will prompt you with increasing line numbers until an empty line (no text before CR) is typed. The line number used as a prompt will be assigned to the line entered by the user in response to it. No error message or warning is issued if existing lines are overwritten while in auto-line mode.

E is invoked from CP/M command mode by typing its name (E) followed by the name of the file you wish to edit. If you specify a file that does not currently exist on the disk, it will be created for you, free of charge! The message 'NEW FILE' will be printed to warn you to check for a mistyped filename if you were intending to edit an existing file. Much like the standard CP/M context editor, ED, the output file will have the same name that you started with, and the original file will be renamed 'x.BAK' where 'x' was the name you started with. Unlike ED, the file will be read into the memory buffer for you. Also unlike ED, you are restricted to files which will fit into your memory along with E and CP/M. If your file is too large, you can always learn to use ED, or split it up into several smaller files.

E will affix line numbers to the lines of your un-numbered file as it reads them in. Likewise, when it writes the file back to disk, it will strip the line numbers from the lines. This allows you to create source programs for translators such as the Microsoft FORTRAN, which will not accept line numbered files. This is also useful for creating documentation files, letters, or whatever.

The command RENUM may be used in place of RESEQ, if desired. Also, either a dash (-) or a comma (,) may be used to separate two arguments (e.g. LIST 10-30 may also be entered as LIST 10,30). An asterisk (*) may be used in place of any number to stand for 65535 (a very big number) - hence to list all lines that contain the word 'data', you might use the command:

```
LIST 0,* /data/
```

A string may be delimited with any character that does not occur in that string, e.g. 'fred', *fred*, or /fred/ are all ok.