MS-DOS KERMIT USER GUIDE

For the IBM PC Family, Compatibles, and Other MS-DOS Systems

Version 2.32

C. Gianone, F. da Cruz

Columbia University Center for Computing Activities
New York, New York 10027

J.R. Doupnik

CASS and EE, Utah State University, Logan, UT 84322

December 11, 1988

1. MS-DOS KERMIT

Program:	Joe R. Doupnik (Utah State University), with contributions by James Harvey (Indiana/Purdue University), James Sturdevant (A.C. Nielson Company), and many others. Originally by Daphne Tzoar and Jeff Damens (Columbia University). See History.
Language:	Microsoft Macro Assembler (MASM)
Version:	2.32
Released:	December 11, 1988.
Documentation:	Christine Gianone, Frank da Cruz (Columbia University), Joe R. Doupnik (Utah State University)
Dedicated To:	Peppi

Kermit-MS Capabilities At A Glance:

Local operation:	Yes
Remote operation:	Yes
Transfers text files:	Yes
Transfers binary files:	Yes
Wildcard send:	Yes
File transfer interruption:	Yes
Filename collision avoidance:	Yes
Can time out:	Yes
8th-bit prefixing:	Yes
Repeat count compression:	Yes
Alternate block check types:	Yes
Terminal emulation:	VT102, H19, VT52, Tektronix 4010
Communication settings:	Speed, Parity, Flow Control, Echo
Transmit BREAK:	Yes (and Long BREAK)
IBM mainframe communication:	Yes
Transaction logging:	Yes
Session logging (raw download):	Yes
Raw upload:	Yes
Act as server:	Yes
Talk to server:	Yes
Advanced server functions:	Yes
Advanced commands for servers:	Yes
Local file management:	Yes
Command/init files:	Yes
Command macros:	Yes
Extended-length packets:	Yes
Local area networks:	Yes (NetBIOS and other support)
MS-Windows compatibility:	Yes
Attribute packets:	Yes
Sliding windows:	No

MS-DOS Kermit, or "Kermit-MS" (or MS-Kermit), is a program that implements the Kermit file transfer protocol for the entire IBM PC family, including the PS/2 series, IBM compatibles, and several other machines based on the Intel 8086 processor series (8088, 80286, 80386, etc) and the DOS operating system family (PC-DOS or MS-DOS, henceforth referred to collectively as MS-DOS or simply DOS).

It is assumed you are acquainted with your PC and with DOS, and that you are familiar with the general ideas of data communication and Kermit file transfer. A very brief overview is given here, but for details consult the early chapters of the *Kermit User Guide* (of which this document is a chapter), or the book Kermit, A File Transfer Protocol, by Frank da Cruz, Digital Press (1987), order number EY-6705E-DP (phone 1-800-343-8321), which also includes background tutorials on computers, file systems, and data communication (including modems, cabling, etc).

For further information about Kermit documentation, updates, lists of current available versions, and ordering information, write to:

Kermit Distribution
Columbia University Center for Computing Activities
612 West 115th Street
New York, NY 10025 (USA)

1.1. System Requirements

Kermit-MS version 2.32 runs in as little as 100K of memory, but will occupy up to 160K or so if it can be found for extra screen rollback memory, macro definitions, etc. Versions not using screen rollback memory will not require the additional space. It will also try to leave 24 Kbytes free for a second copy of `COMMAND.COM` which is needed for execution of certain commands.

On the IBM PC family, Kermit-MS `2.32` performs almost complete emulation of the DEC VT-102 and Heath/Zenith-19 terminals at speeds up to 19,200 baud or greater, lacking only the VT102's smooth scrolling and (on most display boards) 132 column features. And as of version 2.30, Kermit-MS also performs Tektronix 4010/4014 graphics terminal emulation on IBM PC family systems equipped with CGA, EGA, or other graphics adapters, with either color or monochrome monitors.

Much of Kermit's speed is accomplished by direct writes to screen memory, but this is done in a "TopView-aware" manner to allow successful operation in windowing environments like MS-Windows, DesqView, and TopView itself. Speed is also due to direct access of the serial port 8250 UART (Universal Asynchronous Receiver/Transmitter) chip, with buffered, interrupt-driven receipt of characters and selectable XON/XOFF flow control. Full speed 9600 baud operation is possible on 4.77Mhz systems without flow control, but flow control is required on these systems for 19,200 baud or higher rates. The IBM PC version should also run on near-clones like the DG/1 that differ from true PCs only in their choice of UART; non-8250 UARTs are detected automatically, and slower non-interrupt driven Bios serial port i/o is used, in which case the top speed is in the 1200 baud range.

Kermit-MS `2.32` runs on the entire IBM PC family (the PC, XT, AT, PCjr, Portable PC, PC Convertible, PS/2) and compatibles (Compaq, VAXmate, Z150, etc), and there are also specially tailored versions for non-IBM-compatibles like the DEC Rainbow, HP-110, HP-150, HP Portable Plus, Grid Compass II, Victor 9000, and others, plus a "generic DOS" version that should run (slowly) on any 8086-based MS-DOS machine. This document concentrates on the IBM version; some of the system-dependent capabilities described here may be lacking in the non-IBM versions. See section 1.11 for features of different systems.

`KERMIT.EXE` for the IBM PC family occupies about 102K of disk storage (the figure will vary for other versions). This can be reduced by about 15K if you run it through EXEPACK. MS-Kermit is not distributed in packed form, because problems have been reported on certain systems when this is done. So if you decide to pack it, make sure to keep an unpacked version available to fall back to in case of problems.

1.2. History

Over the years, MS-Kermit has grown from a Kermit file transfer program that embodied a simple terminal emulator into a complex and powerful communication program that includes the Kermit file transfer protocol. As a result, the bulk of this manual is devoted to the communication features, rather than Kermit protocol operation. Skip ahead to the next section if you're not interested in the history of MS-Kermit.

MS-DOS Kermit (like the Kermit file transfer protocol itself) is a product of the Systems Group of the Columbia University Center for Computing Activities, and it was one of the four original Kermit programs (with the CP/M, DEC-20, and IBM mainframe versions). It was initially written for the IBM PC with DOS 1.1 by Daphne Tzoar in 1981-1982, based largely on Bill Catchings's original CP/M 8080 assembler version. PC-Kermit (as it was called then) provided basic Kermit file transfer and VT52 emulation. Joellen Windsor of the University of Arizona added conditional assembly support for the Heath/Zenith-100 shortly thereafter, and soon after that Dave King of Carnegie-Mellon University added Heath-19 terminal emulation, and some patches to let the program run under the new DOS version, 2.0. During this era, the program version numbers went from 1.0 to 1.20.

With the appearance in the marketplace of many new MS-DOS machines that were not compatible with the IBM PC, it became apparent that conditionally assembled code supporting each of these machines within a single monolithic source file was not the best way to organize the program. Therefore Daphne, along with Jeff Damens of Columbia, undertook to reorganize the program in a modular way, isolating system dependencies into separate files. The result was version 2.26, released in July 1984. It included support for the DEC Rainbow, the HP-150, the Wang PC, and generic MS-DOS, as well as for the IBM PC family and the H/Z-100. It also included many new features, like 8th-bit prefixing (code contributed by The Source Telecomputing), alternate block check selection, byte-count compression, server/client operation, access to local file and DOS operations, command macros, initialization and command files, screen rollback, key redefinition, and more. For the 2.26 release, the executable Kermit programs were encoded printably as ‘‘`.BOO`’’ files, designed by Bill Catchings as part of this effort, for network and electronic-mail distribution.

Release 2.27 was produced by Daphne and Jeff in December 1984. Unlike 2.26, it ran correctly on the new PC/AT under DOS 3.0, and included support for the NEC APC from Ron Blanford of Seattle, WA, and Ian Gibbons of the University of Hawaii, and for the TI Professional from Joe Smith of the Colorado School of Mines, plus some bug fixes and reorganization. 2.27 is the last version that runs under pre-2.0 versions of DOS.

Version 2.28 (Daphne, Jeff, June 1985) added dynamic memory allocation to reduce disk storage for the `.EXE` file, and to allow the program to adjust itself to the PC’s memory size, plus the inevitable bug fixes (many of them contributed by Edgar Butt of the University of Maryland and Gregg Small of the University of California at Berkeley). During this period, support for additional MS-DOS systems was added by various people.

In December 1985, a tape showed up at Columbia sent by Prof. Joe R. Doupnik of the Center for Atmospheric and Space Studies and EE Department at Utah State University. This tape contained version 2.28 modified to fully support the DOS 2.0 file system, and to which many new features had been added, notably the ability of the MS-DOS Kermit server to process various REMOTE commands (DIR, CWD, SPACE, etc). And at about the same time, a tape arrived from James Harvey of Indiana/Purdue University, who had changed Kermit’s CONNECT command to emulate the popular DEC VT100 terminal. James’s material was sent to Joe, who then laboriously fitted the VT100 emulation into his own code, keeping the VT52 and H19 emulation alive as options, and upgrading the VT100 emulation to VT102 by adding features such as line and character insertion and deletion. The result was version 2.29, released in May 1986.

Soon after the release of 2.29, some disks were sent in by James Sturdevant of the A.C. Nielson Company, containing a full implementation of the Kermit script facility, as described in the Kermit book. This material was sent to Joe, who had by now become keeper of MS-DOS Kermit and had already begun work on version 2.30 by adding support for extended-length packets. Joe had been carrying on voluminous network correspondence (Thanks, BITNET!) with Columbia and with MS-DOS Kermit users and testers all over the world, giving birth to many new features, including Tektronix graphics terminal emulation, support for operation over local area networks, support for 8-bit ASCII terminal connections and international character sets, ANSI printer control, and a redesigned, more powerful, more portable key redefinition mechanism.

Version 2.30 was formally released on January 1, 1988, after many "alpha" and "beta" tests. Among the many contributors to this version were Brian Holley and Joe Smith for the Tektronix emulation, Robert Goeke for the NEC AP3 support, Brian Peterson and Andreas Stumpf for the Victor 9000, Bob Babcock and Joe White for the Sanyos, Christopher Lent for the Wang PC, Jack Bryans for an Intel iRMX version, Jim Noble for the Grid Compass, Geoff Mulligan and others for the Zenith 100, and David Knoell for the special Rainbow edition. And thanks to Gisbert Selke, Jack Bryans, and others for proofreading drafts of this manual, with apologies to anyone we neglected to mention.

Work on version 2.31 began within weeks of the release of 2.30. The major new features were an improved command interface, a fully capable script programming language, and inclusion of file attributes packets to send the time, date and size of files along with the data. Support for Ungermann-Bass Net One LAN was also added, thanks to contributions from Henrik Levkowetz and Renne Rehmann. These changes led to a fairly thorough revision of the interior while providing the familiar commands and new features. Meanwhile, Horofumi Fujii and Akihiro Shirahasi of the National Laboratory for High-Energy Physics (KEK) in Japan adapted 2.31 to the NEC PC-9801, and for this machine added support for Japanese Kana and Kanji character sets.

Version 2.32 was issued by Joe in December 1988. It included the usual bug fixes, plus several new script programming features, and improved support for international use, allowing for languages like Hebrew and Arabic that print right to left, adapted from work by Baruch Cochavy, IIT, Technion, Haifa, Israel. Thanks also to Glenn Trewitt, Mark Zinzow, and Ken Ridley for valuable suggestions and contributions to this release.

Like all Kermit programs, MS-DOS Kermit may be freely copied and shared, so long as it is not done for profit.

1.3. Using MS-Kermit

MS-DOS Kermit performs two major functions, terminal emulation and file transfer. File transfer can be done using either the Kermit file transfer protocol, or else (without error checking), ASCII or XON/XOFF capture and transmission methods. To use Kermit for "raw" uploading or downloading of files, see the descriptions of the TRANSMIT and LOG SESSION commands.

Before you can transfer files with another system using Kermit protocol, you must first connect to it as a terminal, login if necessary, and start up a Kermit program there. Kermit's CONNECT command lets you do this by making your PC act like a terminal. After setting things up on the other computer, you must return to the PC and tell it what to do. Returning to the PC is accomplished by typing a special sequence of characters, called the "escape sequence."

The following example shows this process; the other computer is a Unix system, but the method is the same with most others. The parts you type are underlined (if this document was printed on a printer that can underline), and when you type a command, you terminate it with a carriage return, which you can't see in the example. The mysterious ``^]c'' is MS-Kermit's escape sequence, which you enter by holding down the Control (Ctrl) key and pressing ``]'' (right square bracket), and then typing the letter C. The example assumes the MS-Kermit program is stored on disk as `KERMIT.EXE`.

```
Program Dialog:                          Explanation:

A>kermit
IBM PC Kermit-MS V2.32 11 Dec 1988       Program's greeting.
Type ? or HELP for help

Kermit-MS>set speed 1200                 Set the right baud rate.
Kermit-MS>connect                        Connect as a terminal.
ATDT7654321                              Dial the modem if necessary.
CONNECT 1200                             The modem tells you you're connected.

    Now you're talking to the Unix system.
    Type a carriage return to get its attention.

Login: max                               Login to the host.
password:______                          (Passwords normally don't echo.)
% kermit                                 Run Kermit on the host.
C-Kermit>receive                         Tell it to receive a file.
^]c                                      Escape back to the PC.
Kermit-MS>send autoexec.bat              Send a file.

    (The file is transferred . . . )

Kermit-MS>                               Transfer complete, prompt reappears.
```

In this example, the user types "kermit", and sees the program's herald and its prompt, ‘‘`Kermit-MS>`’’. Then she sets the appropriate communication speed ("baud rate"), connects as a terminal, issues a dialing command to a Hayes-like modem (you would skip this step if you had a direct connection), logs in to her ID on the Unix system which she has dialed, starts "C-Kermit" on the Unix system, tells it to receive a file, escapes back to the PC, and tells MS-Kermit to send a file. After the file is transferred, the user would normally connect back to the Unix system, exit from the Kermit program there, and log out:

```
Kermit-MS>connect               Connect again.
C-Kermit>exit
% ^D                            Logout from Unix by typing Ctrl-D.
^]c                             Escape back to the PC.
Kermit-MS>exit                  Return to DOS.
```

To transfer a file in the other direction, simply exchange the "send" and "receive" commands above. That's the easiest and quickest way to use Kermit. If this simple scenario does not work for you, issue the MS-Kermit SHOW COMMUNICATIONS command and look for any obvious incorrect settings (port, speed, parity), fix them with SET commands (described in Section 1.6.10), and try again. (IBM mainframe linemode connections have so many "different" settings, there's a special command to do them all at once, "do ibm", which you would type as the first Kermit-MS command above.) If that doesn't help, read on. Many problems can crop up when you attempt to connect two unlike systems over a possibly hostile communication medium. And if you intend to be a frequent user of Kermit, there are many options you can take advantage of to adapt MS-Kermit to different systems, improve its performance, and automate common tasks.

1.4. The MS-DOS File System

The features of the MS-DOS file system of greatest interest to Kermit users are the form of the file specifications, and the formats of the files themselves.

1.4.1. File Specifications

MS-DOS file specifications (in version 2.0 or later of DOS) are of the form

```
DEVICE:\PATHNAME\NAME.TYPE
```

where the DEVICE is a single character identifier (for instance, A for the first floppy disk, C for the first fixed disk, D for a RAM disk emulator) followed by a colon (‘‘`:`’’), PATHNAME is up to 63 characters of identifier(s) (up to 8 characters each) surrounded by backslashes (‘‘`\`’’), NAME is an identifier of up to 8 characters, and TYPE is an identifier of up to 3 characters in length. Device and pathname may be omitted. The first backslash in the pathname may be omitted if the specified path is relative to the current directory. In the path field, ‘‘`.`’’ means the current directory, ‘‘`..`’’ means the parent directory. Some DOS implementations (like Wang) may use slash (‘‘`/`’’) rather than backslash as a directory separator.

Pathname is normally omitted, but can be specified in all Kermit-MS commands (as of version `2.29`). Device and directory pathnames, when omitted, default to either the user's current disk and directory, or to the current directory search path as specified in the DOS PATH environment variable, depending on the context in which the file name appears.

> When this document says that a file is searched for "in the current path," it means that Kermit-MS looks on the current disk and directory first, and if the file is not found, then the directories listed in the PATH environment variable are searched. If the PATH environment variable is empty, Kermit looks only at the current disk and directory.

`NAME.TYPE` is sufficient to specify a file on the current disk and directory, and only this information is sent along by Kermit-MS with an outgoing file.

The device, path, name, and type fields may contain uppercase letters, digits, and the special characters ‘‘`-`’’ (dash), ‘‘`_`’’ (underscore), ‘‘`$`’’ (dollar sign), ‘‘`&`’’ (ampersand), ‘‘`#`’’ (number sign), ‘‘`@`’’ (at sign), ‘‘`!`’’ (exclamation mark), ‘‘'’’ (single quote), ‘‘()’’ (parentheses), ‘‘{}’’ (curly braces), ‘‘^’’ (caret or circumflex), ‘‘~’’ (tilde), and ‘‘`’’ (accent grave). Normally, you should confine your filenames to letters and digits for maximum transportability to non-DOS systems. When you type lowercase letters in filenames, they are converted automatically to uppercase. There are no imbedded or trailing spaces. Other characters may not be included; there is no mechanism for "quoting" otherwise illegal characters in filenames. The fields of the file specification are set off from one another by the punctuation indicated above.

The name field is the primary identifier for the file. The type, also called the extension or suffix, is an indicator which, by convention, tells what kind of file we have. For instance `FOO.BAS` is the source of a BASIC program named FOO; `FOO.OBJ` might be the relocatable object module produced by compiling `FOO.BAS`; `FOO.EXE` could be an executable program produced by loading `FOO.OBJ`, and so forth. `.EXE` and `.COM` are the normal suffixes for executable programs.

MS-DOS allows a group of files to be specified in a single file specification by including the special "wildcard" characters, ‘‘*’’ and ‘‘?’’. A ‘‘*’’ matches any string of characters from the current position to the end of the field, including no characters at all; a ‘‘?’’ matches any single character. Here are some examples:

`*.BAS` All files of type `BAS` (BASIC source files) in the current directory.

`FOO.*` Files of all types with name `FOO`.

`F*.*` All files whose names start with F.

`*.?` All files whose types are exactly one character long, or have no type at all.

Wildcard notation is used on many computer systems in similar ways, and it is the mechanism most commonly used to instruct Kermit to send a group of files.

Users of Kermit-MS should bear in mind that other (non-MS-DOS) systems may use different wildcard characters. For instance VMS and the DEC-20 use ‘‘%’’ instead of ‘‘?’’ as the single character wildcard; when using Kermit-MS to request a wildcard file group from a Kermit-20 server, the DOS ‘‘?’’ must be replaced by the DEC-20 ‘‘%’’.

1.4.2. File Formats

MS-DOS systems store files as streams of 8-bit bytes, with no particular distinction among text, program code, and binary files. ASCII text files consist of lines separated by carriage-return-linefeed sequences (CRLFs), and this conforms exactly to the way Kermit represents text files during transmission, so Kermit-MS has no need for a SET FILE TYPE BINARY command. But since a non-MS-DOS receiving system might need to make distinctions as to file type, you will probably have to issue SET FILE TYPE commands there if you are sending it non-text files. In transmitting files between Kermit-MS programs, regardless of file contents, the receiving MS-DOS system is equally capable of processing text, code, and data, and in fact requires no knowledge of how the bytes in the file are to be used.

MS-DOS (unlike CP/M) knows the exact end of a file because it keeps a byte count in the directory, so one would expect no particular confusion in this regard. However, certain MS-DOS programs continue to use the CP/M convention of terminating a text file with a Control-Z character, and won’t operate correctly unless this terminating byte is present. Therefore, you should be aware of a special SET EOF option for both incoming and outbound files, described later.

Non-MS-DOS systems may be confused by nonstandard ASCII files sent by Kermit-MS:

- Files containing any of the 8-bit "extended ASCII" characters may need conversion (or translation) to 7-bit ASCII.

- Files produced by word processing programs like Word Perfect or Word Star may contain special binary formatting codes, and could need conversion to conventional 7-bit ASCII format prior to transmission, using an "export" procedure.

- Files created by word processors that store formatting data at the end of the file, after the Control-Z and before physical end, may require special processing via SET EOF to strip the formatting data, lest they confuse non-MS-DOS recipients.

- Spreadsheet or database files usually need special formatting to be meaningful to non-MS-DOS recipients (though they can be transmitted between MS-DOS systems with Kermit-MS). Such programs usually come with an "export" procedure to convert their files to plain ASCII text.

- BASIC programs are normally saved in a binary "tokenized" form. Use BASIC's ‘‘`,a`’’ SAVE option to save them as regular ASCII text, as in

```
save"foofa",a
```

In general, when attempting to transfer non-text files between MS-DOS and a different kind of system, consult the Kermit manual for that system.

1.5. Program Setup and Invocation

The MS-DOS Kermit program can be run from any disk without any special installation procedure. On hard disk systems, it is convenient to store the program in one of the directories listed in your DOS PATH, and it is often desirable to customize Kermit's operation to your communications and computing environment by creating an initialization file.

Kermit-MS can be run interactively, from a batch file, as an "external" DOS command, or from redirected standard input. Commands consist of one or more fields, separated by "whitespace" -- one or more spaces or tabs.

Upon initial startup, the program executes any commands found in the file `MSKERMIT.INI` on the current disk, or (if not found on the current disk) in the first directory containing a file by that name, from the list in your DOS PATH environment variable. The Kermit initialization file may contain command macro definitions, communications settings for one or more ports, or any other Kermit-MS commands, and you may create it using any text editor capable of saving files in plain ASCII text format. Here is a sample:

```
comment -- MSKERMIT.INI, MS-DOS Kermit initialization file

comment -- Don't overwrite my files!
 set warning on

comment -- Define macros for the systems I use...
 define unix set local-echo off,set par non,set flow xon,set timer off
 def ibm set par odd,set loc on,set hands xon,set flo none,set tim on
 def modem set port 2, set speed 1200

comment -- Define macros for quickly adapting to varying
 def noisy  set block-check 3, set receive packet 40, set retry 20
 def normal set block-check 1, set rec pack 94, set retry 5
 def clean  set block-check 2, set rec pack 500, set retry 5

comment -- I always start out by connecting to my UNIX system...
 set port 1
 set speed 4800
 do unix
 connect
```

A different file may be substituted for `MSKERMIT.INI` by using "`-f` *filename*" on the DOS command line, e.g.

```
kermit -f monday.ini
```

The meanings of these commands will emerge below. For now, just note how you can use command files (and "macro definitions") to easily adapt MS-Kermit to widely differing communication environments. A more advanced initialization file is shown in section 1.9.

Interactive Operation:

To run Kermit-MS interactively, invoke the program from DOS command level by typing its name, normally "kermit" (this means the program should be stored in your path with the name `KERMIT.EXE`). When you see the program's prompt,

```
Kermit-MS>
```

you may type Kermit commands repeatedly until you are ready to exit the program, as in the following example (which assumes there's already a Kermit "server" set up on the other end):

```
A>
A>kermit

IBM PC Kermit-MS V2.32  11 Dec 1988
Type ? or HELP for help

Kermit-MS>set speed 19200
Kermit-MS>send foo.*
```

The files are sent.

```
Kermit-MS>get fot.*
```

The requested files are received.

```
Kermit-MS>exit
A>
```

Interactive commands are described in Section 1.6.

Command Line Invocation:

Kermit-MS may be invoked with command line arguments from DOS command level, for instance:

```
A>kermit send peter.amy
```

or

```
A>kermit set port 1, set speed 9600, connect
```

In this case, help and completion are not available (because the program that provides them won't start running until after you type the entire command line), and Kermit-MS will exit back to DOS after completing the specified command or commands. Therefore, when invoked with command line arguments, Kermit-MS will behave as if it were an external DOS command, like MODE. Note that several commands may be given on the command line, separated by commas. This can't be done interactively or from TAKE command files.

Two special Kermit commands can be given on the DOS command line. First is the keyword STAY which prevents Kermit from exiting naturally when the last command has completed (unless, of course, EXIT or QUIT was among the commands). The second command is

`-F` *filename*

This means use the indicated filename as the initialization file rather than `MSKERMIT.INI`. The PATH will be searched for this file, if necessary. A space or tab must separate -F from the filename, and the F may be in upper or lower case. Example:

```
kermit -f tuesday.ini, set port 2, do ibm, stay
```

You can run Kermit with no initialization file at all by using the command

```
kermit -f nul
```

If `-F` is the only command line option, STAY is implied.

Redirected Input and Output

Kermit-MS also can be operated by redirecting input to it from a file, as in:

```
C>kermit < myscript.txt > myscript.log
```

or from a DOS "pipe", as in

```
C>sort < sends.txt | kermit
```

The file `MYSCRIPT.TXT` contains Kermit commands as if they were typed manually. The DOS symbol "<" means that Kermit should read from the following file rather from the keyboard.

Kermit knows this is occurring and takes special steps to avoid the real keyboard and to quit when the file has been completely examined. The filename can also be the name of a device, such as COM1, to converse on the same or different line as file transfer traffic. Information destined for the screen still goes to the screen unless the phrase "> *filespec*" is added to the command line above to send the normal screen output to a file or device (device COM1 also works). Note that the terminal emulation screen cannot be redirected.

Batch Operation:

Like many other MS-DOS programs, Kermit-MS may be operated under DOS batch with command line arguments. If you invoke it without command line arguments, it will run interactively, reading commands from the keyboard and not the batch file. When it exits, batch processing will continue to the end of the batch file.

Kermit-MS returns the "errorlevel" parameter used as program exit status. Present values are in the range 0 to 7 with three areas yielding success or failure reports for the entire Kermit session. The errorlevel values are:

errorlevel	Kermit session status
0	entirely successful operation
1	a Send command completed unsuccessfully
2	a Receive or GET command completed unsuccessfully
4	a REMOTE command completed unsuccessfully
3,5,6,7	combinations (addition) of the above conditions

Note that failures are remembered for the whole session and are not canceled by a following successful operation of the same type. Thus, sending several files individually yields an errorlevel of 0 only if all the files were sent successfully. The "errorlevel" parameter also applies to script commands where OUTPUT corresponds to SEND and INPUT to RECEIVE. An example of Batch invocation of Kermit is shown in Figure 1-4.

You may also force Kermit to return any desired errorlevel, using the SET ERRORLEVEL command. DOS batch parameters may be passed along to Kermit; see section 1.7 for details.

Remote Operation:

The MS-DOS CTTY command allows an MS-DOS system to be used from a terminal connected to its communication port. Such sessions must be conducted with great care, since many programs assume that they are running on the real console, and explicitly reference screen memory or the physical keyboard. Kermit can be used in this manner too, but before you give it any file transfer commands, you must inform it that it is running in "remote mode" rather than its normal "local mode." Use the SET REMOTE ON command for this purpose, to prevent the file transfer display from being sent out the port.

RAM Disk Operation:

If you invoke Kermit frequently, and you have sufficient memory on your PC, you may find it convenient to copy Kermit and its initialization file to a RAM disk when you start your system. This allows Kermit to be started and used quickly and silently, with no mechanical disk operations.

For instance, if you're using IBM's VDISK facility to create the RAM disk, you might put statements like this in your `CONFIG.SYS` file:

```
DEVICE=VDISK.SYS 384 512 128 /e
```

This assumes you have 384K of extended (/e) memory installed and VDISK.SYS is in the root directory of the boot disk. It creates a 384K RAM disk with 512B sector size and space for 128 directories in the extended memory, assigning it the disk letter of your first unused disk. And then in your AUTOEXEC.BAT file (assuming the RAM disk is disk D:) . . .

```
COPY KERMIT.EXE   D: >NUL
COPY MSKERMIT.INI D: >NUL
COPY COMMAND.COM  D: >NUL
SET COMSPEC=D:\COMMAND.COM
PATH D:\; ...
```

The PATH command allows DOS to find KERMIT.EXE, and Kermit to find MSKERMIT.INI and COMMAND.COM, on the RAM disk. If you use Kermit transfer files to your RAM disk, remember to copy those files to a real disk before you turn off the system.

Use of MS-Kermit in Windowing and Multiprocessing Environments:

Kermit-MS can operate within windowing environments like such as TopView, DESqview, and MS-Windows. It runs in an active window under MS-Windows, accepts cut and paste material, talks with mice, and shrinks to an icon (a boxed "KER"). An MS-Windows .PIF file can be constructed for Kermit using the PIFEDIT program, supplied with Windows. Memory requirements should be listed as 102 to 160KB. It should state that Kermit does not modify the screen, keyboard, memory, COM1, or COM2 (not true but it satisfies Windows). Program switch and exchange should be marked as Text, and Close Window on Exit should be checked. This configuration will let you run Kermit with all the Windows features, but slowly. To run at full speed under Windows, tell PIFEDIT that Kermit modifies the screen. Then you lose the Windows features (cutting, pasting, running the clock at the same time, etc), but you still get back to the Windows interface when you EXIT Kermit.

MS-Kermit has also been reported to operate successfully under Concurrent DOS. However, since it does not interact explicitly with the Concurrent DOS time-slice scheduler, Kermit will tend use a lot of CPU cycles.

Local Area Network Operation:

MS-Kermit is capable of using a serial port on another local area network (LAN) node, so long as that node is running an asynchronous communication server and you have installed a device driver on your own PC that makes COM1 or other communication port i/o use the network server. This type of connection works because MS-Kermit 2.30 and later releases on IBM PCs check the selected port to see if it's a real 8250 UART chip, and if it isn't, Kermit uses only Bios calls for port i/o, and the network routes these through your network device driver. It may be desirable to give the command SET PORT BIOS*n* (*n* is a digit 1-4) to actively select the Bios port rather than a real hardware device. This style of operation should be transparent to Kermit, but not all asynchronous communications servers utilize this technique.

As of version 2.30, the IBM PC version of Kermit can also communicate directly with another PC on a local area network through the IBM NetBIOS emulator distributed with the LAN. In essence, the LAN substitutes for the serial port, modem, and other wiring. Kermit running on one user machine can transfer files with another Kermit also on the network much as if they were connected by modems, and Kermit can talk with some larger machines the same way. The important network command is

    SET PORT NETBIOS *nodename*

for NetBios, or

    SET PORT UB-NET1 *nodename*

for Ungermann-Bass Net-One NETCI. For details, see the description of the SET PORT and SERVER commands, and (if you're interested) Section 1.18.1 for a technical description.

Kermit can even communicate with some other computers, such as Unix systems, which accept logins via this remote pathway. The initial startup is the same as calling a mainframe and logging in except the command SET

PORT NET *nodename* is used instead of SET PORT COM1. A connection is established with the first use of the communications circuit, such as CONNECT, REMOTE DIR, SEND, or other file transfer command, and terminated with the HANGUP command.

1.6. Kermit-MS Commands

MS-DOS Kermit has the following commands:

-F	specify alternate init file name on DOS command line.
ASK	user to type text, in response to a prompt.
ASSIGN	the value of one variable to another.
BYE	to remote server, exit from MS-Kermit.
CLEAR	serial port buffer.
CLOSE	log files and stop logging remote session.
COMMENT	For including comments in command files.
CONNECT	as terminal to remote system (C).
CWD or CD	change local working directory.
DEFINE	a macro of Kermit-MS commands.
DELETE	local files.
DIRECTORY	listing of local files.
DISABLE	server recognition of selected commands.
DO	a command macro.
ECHO	a line of text on the screen.
ENABLE	server recognition of selected commands.
EXIT	from Kermit-MS.
FINISH	Shut down a remote Kermit server.
GET	remote files from server.
GOTO	jump to labeled line in script file.
HANGUP	the phone or network connection.
HELP	about Kermit-MS.
IF	decision-making in Take or Macro scripts.
INPUT	specified string from serial port, for scripts.
LOG	remote terminal session, transactions, or packets.
LOGOUT	remote server, don’t exit from Kermit-MS.
MAIL	send file to remote Mailer via Kermit.
OUTPUT	string out serial port, for scripts.
PAUSE	between commands.
POP	exit Take file or Macro.
PUSH	to MS-DOS command level.
QUIT	from Kermit-MS (same as EXIT).
RECEIVE	files from remote Kermit (R).
REINPUT	reread script Input buffer.
REMOTE	Prefix for remote file management commands.
RUN	an MS-DOS program or command.
SEND	files to remote Kermit (S).
SERVER	mode of remote operation.
SET	various parameters.
SHOW	various parameters.
SPACE	inquiry (about disk space).
STATUS	inquiry (about settings).
STAY	stay within Kermit after DOS command line invocation.
STOP	exit all Take files or Macros.
TAKE	commands from a file.
TRANSMIT	a file "raw" (no error checking).
TYPE	a local file on the screen.
VERSION	display Kermit-MS program version number.
WAIT	for the specified modem signal to appear.

Not all of these commands are necessarily available on all MS-DOS systems, and some of the commands may work somewhat differently between DOS versions.

A command keyword, such as SEND, RECEIVE, HELP, etc, may be abbreviated, so long as you have typed enough letters to distinguish it from other keywords that are valid in that position. For instance, you can type CLE for CLEAR and CLO for CLOSE. Several common commands also have special non-unique abbreviations, like C for CONNECT, S for SEND, and R for RECEIVE. Kermit will notify you if you have typed a word with too few letters.

During interactive operation, you may edit the command you're currently typing using BACKSPACE to erase the character most recently typed, Ctrl-W to delete the most recent field, or Ctrl-U to delete the entire command. The editing characters may be used in any combination until the command is finally entered by typing RETURN (Carriage Return, Enter) or Ctrl-L.

You may use the help (``?'') and keyword completion (ESC) features freely while typing Kermit-MS commands. A question mark typed at almost any point in a command produces a brief description, or "menu", of what is expected or possible at that point. ESC typed at any point, except in a local filename, will cause the current field to be filled out if what you have typed so far is sufficient to identify it, and will leave you in position to type the next field (or to type a ``?'' to find out what the next field is); otherwise, the program will beep at you and wait for you to type more characters.

As of version 2.31, Kermit-MS recognizes full 8-bit character inputs, with only NUL, ESC, DEL/BS, Ctrl-W (delete word), Ctrl-U (delete line), and Ctrl-C being special. This is to enhance support for various languages and keyboards. The SET KEY and SHOW KEY commands can prompt for keyboard input and understand 8-bit characters but only at their interactive prompt. The SET KEY, INPUT, and OUTPUT commands accept "backslash number format" on the main Kermit command line. Thus, national characters which are full 8-bit codes can be expressed on command lines in backslash number form (\ddd), provided the Kermit command itself can understand the form. Most commands that want numbers or single characters as operands understand this notation. To enter characters in backslash number format, type a backslash (``\'') followed by a number corresponding to the ASCII code for the character. MS-Kermit accepts many different backslash codes in different contexts. These are summarized in Table 1-1; letters following the backslach may be either upper or lower case.

`\123`	(up to 3 decimal digits) - A decimal number
`\d123`	(up to 3 decimal digits) - A decimal number
`\o123`	(up to 3 octal digits) - An octal (base 8) number
`\x123`	(up to 3 hexadecimal digits) - a hexadecimal (base 16) number
`\{ }`	For grouping, e.g. `\{12}6 = Ctrl-L 6`, not ~
`\;`	Include a semicolon in a TAKE-file command or macro definition.
`\%`	Introduce a Kermit variable, `\%1, \%2, ..., \%a, \%b, ... \%z`
`\K`	A Kermit connect-mode verb like `\Kexit` (see Table 1-6)
`\B`	Send a BREAK (OUTPUT command only)
`\255`	Shorthand for CRLF or LFCR (INPUT command only)
`\CD`	Carrier Detect RS-232 signal (WAIT command only)
`\DSR`	Data Set Ready RS-232 signal (WAIT command only)
`\CTS`	Clear to Send RS-232 signal (WAIT command only)

Table 1-1: MS-DOS Kermit Backslash Codes

Table 1-2 shows all of the 7-bit ASCII codes in decimal. Most Kermit commands understand backslash-ASCII codes, both imbedded within character strings, and alone, as when a single character or number is to be specified.

Some Kermit-MS commands like GET, SHOW KEY, and SET KEY, may prompt for additional information on

Dec	Name	Ctrl	Dec	Char	Dec	Char	Dec	Char
0	NUL	^@	32	SP	64	@	96	‘
1	SOH	^A	33	!	65	A	97	a
2	STX	^B	34	"	66	B	98	b
3	ETX	^C	35	#	67	C	99	c
4	EOT	^D	36	$	68	D	100	d
5	ENQ	^E	37	%	69	E	101	e
6	ACK	^F	38	&	70	F	102	f
7	BEL	^G beep	39	’	71	G	103	g
8	BS	^H backspace	40	(	72	H	104	h
9	HT	^I tab	41	)	73	I	105	i
10	LF	^J linefeed	42	*	74	J	106	j
11	VT	^K	43	+	75	K	107	k
12	FF	^L formfeed	44	,	76	L	108	l
13	CR	^M return	45	-	77	M	109	m
14	SO	^N shift out	46	.	78	N	110	n
15	SI	^O shift in	47	/	79	O	111	o
16	DLE	^P	48	0	80	P	112	p
17	DC1	^Q XON	49	1	81	Q	113	q
18	DC2	^R	50	2	82	R	114	r
19	DC3	^S XOFF	51	3	83	S	115	s
20	DC4	^T	52	4	84	T	116	t
21	NAK	^U	53	5	85	U	117	u
23	ETB	^W	54	6	86	V	118	v
22	SYN	^V	55	7	87	W	119	w
24	CAN	^X	56	8	88	X	120	x
25	EM	^Y	57	9	89	Y	121	y
26	SUB	^Z	58	:	90	Z	122	z
27	ESC	^[escape	59	;	91	[	123	{
28	FS	^\	60	<	92	\	124	|
29	GS	^]	61	=	93	]	125	}
30	RS	^^	62	>	94	^	126	~
31	US	^_	63	?	95	_	127	RUBOUT,DELETE

Table 1-2: The US ASCII Character Set (ANSI X3.4-1977)

subsequent lines. If you have reached one of these prompts and then wish to cancel the command, you may type Control-C to get back to the main `Kermit-MS>` prompt.

Summary of Kermit-MS command editing characters:

SPACE Separates fields within the command.

TAB Same as Space, and echoes as Space. You may also use Ctrl-I for Tab.

BACKSPACE
Deletes the character most recently typed. May be typed repeatedly to delete all the way back to the prompt. You may also use DELETE, RUBOUT, Ctrl-H, or equivalent keys.

Ctrl-W Deletes the most recent "word", or field, on the command line. May be typed repeatedly.

Ctrl-U Deletes the entire command line, back to the prompt.

Ctrl-C Cancels the current command and returns to the "`Kermit-MS>`" prompt. Also, terminates execution of a TAKE command file.

ESC If enough characters have been supplied in the current keyword to identify it uniquely the remainder of the field is supplied and the cursor is positioned to the next field of the command. Otherwise, a beep is sounded. ESC does not provide filename completion.

- `?` Displays a brief message describing what may be typed in the current command field. Also, wildcard character for matching any single character in all but the first position of a filename.
- `#` Wildcard character for matching single characters in filenames. Equivalent to MS-DOS ‘‘`?`’’, but used in the first position of a filename only, so that ‘‘`?`’’ may be used to get help at the beginning of a filename field.
- ENTER Enters the command. On most keyboards, you may also use RETURN or Ctrl-M.
- Ctrl-L Clears the screen and enters the command.

Liberal use of ‘‘`?`’’ allows you to feel your way through the commands and their fields. This feature is sometimes called "menu on demand" or "context sensitive help" -- unlike systems that force you to negotiate menus at every turn, menu-on-demand provides help only when it is needed.

Command reading is done through DOS calls and Kermit key redefinition does not apply at Kermit-MS command level. But `ANSI.SYS` or other external console drivers can be used for this purpose, for instance to assign ESC to the PC’s backquote key (`ANSI.SYS` is the IBM-supplied extended screen and keyboard device driver, described in the IBM DOS Technical Reference Manual). Other console drivers available include ProKey, SuperKey, `NANSI.SYS` (a public-domain replacement for `ANSI.SYS`), and FANSICONSOLE.

The notation used in command descriptions is as follows:

[square brackets]
: An optional field. This field may be omitted.

{curly braces}
: A list of alternatives, separated by commas. Choose one of the items from the list.

italics
: Shows parameters, such as numbers or filenames, are shown in italics (providing the printer is capable of printing italics). You substitute the actual number or filename.

<u>underlining</u>
: In dialog examples, the characters you should type are underlined (on printers that can show it) to distinguish them from computer typeout.

`hh:mm:ss`
: A time of day, in 24-hour notation (10:00:00 is 10 AM; 23:30:00 is 11:30 PM), which may not be more than 12 hours later than the current time.

The following sections describe all the MS-DOS Kermit commands. Since some command descriptions may contain references to other commands that haven’t been explained yet, you might find that this manual makes more sense on a second reading.

1.6.1. Program Management Commands

"Program management" is a rubric for Kermit-MS commands like TAKE, EXIT, HELP, COMMENT, ECHO, and VERSION, that don’t fall into any other category.

HELP displays a one screen introduction to frequently used Kermit commands and their editing keys, and suggests using the question mark command to see the terse list of primary level Kermit commands.

VERSION displays the MS-Kermit program version number, which you should know in case you are reporting bugs or seeking technical assistance.

Other program management commands require a bit more explanation.

The EXIT Command

Syntax: `EXIT` *or* `QUIT`

EXIT and QUIT are synonyms for each other. They cause MS-Kermit to return control to DOS or whatever program invoked MS-Kermit. The specific actions taken are:

- Close any open log or other files.
- Close any open network connection.
- Release all memory claimed by the program.
- Return interrupts for the currently selected communication device to their original owner.
- Terminate execution.

The serial port RS-232 signals are left alone upon EXIT, so that modem connections are not broken. Kermit-MS may be restarted with the connection intact. Use HANGUP to explicitly break a modem connection; and use SHOW MODEM or SHOW COMMUNICATIONS to view the status of modem signals CD (Carrier Detect), Data Set (modem) Ready (DSR), and Clear To Send (CTS).

The STAY Command

Syntax: `STAY`

The STAY command, if included among command line arguments, instructs MS-Kermit not to exit upon completion but rather to enter interactive mode, unless EXIT or QUIT was among the command arguments. STAY has no effect when entered interactively or from a TAKE file.

The PUSH Command

Syntax: `PUSH`

PUSH is similar to EXIT, except it leaves MS-Kermit intact by invoking an MS-DOS command processor "under" Kermit-MS, either `COMMAND.COM` or whatever shell you have specified with COMSPEC (or SHELL, depending on the system) in your `CONFIG.SYS` file. You can return to Kermit-MS by typing the MS-DOS EXIT command, and you will find Kermit-MS as you left it, with all settings and the terminal emulation screen intact. The same function is invoked by the CONNECT escape-level command P. Example:

```
Kermit-MS>push                  Push to DOS.
Command v. 3.30                 COMMAND.COM program herald.

C>diskcopy a: b:                Run a DOS program.

   DISKCOPY dialog here . . .

C>dir b:                        More DOS commands . . .

   DOS session continues . . .

C>exit                          When done, type DOS EXIT command.
Kermit-MS>                      Back at Kermit.
```

The TAKE Command

Syntax: `TAKE` *filespec*

The TAKE command gives you way a to collect MS-Kermit commands into a single file, so that you can execute many commands by typing a single (TAKE) command. TAKE instructs MS-Kermit to execute commands from the file that you specify. The current directory is searched for the file first, and then any directories listed in the PATH environment variable. The command file may include any valid Kermit-MS commands, including TAKE, but it cannot include characters to be sent to a remote host after a CONNECT command (use scripts for that, described below). Execution of a TAKE file may be cancelled by typing Control-C at the keyboard.

An implicit TAKE command is executed upon the initialization file, `MSKERMIT.INI` (or another file specified in the ‘‘-f’’ command-line argument), whenever you start MS-Kermit. The MSKERMIT.INI file contains any commands you want to be executed each time you run Kermit. A sample is shown above, and a more ambitious example is shown in section 1.9.

Commands within TAKE files, unlike interactive commands, may include trailing comments, preceded by semicolons:

```
set port 2      ; Select the modem port.
set speed 1200  ; Set the baud rate for the modem.
connect         ; Conduct a terminal session.
hangup          ; Hang up the phone after escaping back.
```

Note the HANGUP command after CONNECT. The HANGUP command is not executed until after you escape back from your CONNECT session. If this file were called MODEM.CMD, the following TAKE command would execute it:

```
Kermit-MS>take modem.cmd
```

This directs MS-Kermit to find the MODEM.CMD file, open it, execute the commands in it, close it, and return to the MS-Kermit> prompt when done. This process can take a while on floppy-disk based systems.

Since TAKE file processing discards all characters from a line beginning with the first semicolon, it is normally not possible to include semicolons as part of the commands themselves, e.g.

```
get dska:foo.bar;6
```

To get around this restriction, you may precede such semicolons with a backslash:

```
get dska:foo.bar\;6
```

Commands from the TAKE file will normally not be displayed on your screen during execution. If you want to see them as they are executing, you can SET TAKE-ECHO ON (for instance, at the beginning or end of your MSKERMIT.INI file). With the echoing ON, comments are also displayed for reference, but the semicolon is not shown.

TAKE files may be nested to a reasonable level. A command file that was invoked by another command file normally returns to its invoking command file, rather than to the MS-Kermit> prompt, when the end of the command file is reached.

TAKE files have two commands to quit processing before the end of the file is reached. The POP command exits the current TAKE file (or macro) and returns control to the previously executing TAKE or macro, where one is invoked within another. The STOP command exits all TAKE files and macros and returns directly to the Kermit prompt.

In TAKE files (and macro definitions, which are discussed later), long commands may be continued on subsequent lines by terminating each continued line with a hyphen (minus sign). If a line needs to terminate with a real minus sign it may be expressed numerically as \45 or can be extented with extra spaces. The overall command length is normally 127 bytes (a beep sounds near this limit).

An explicit question mark (‘‘?’’) in a TAKE file will cause a help message to be displayed and the rest of the line will be read as another command. If you need to include a question mark in a command, use the ASCII backslash notation "\63".

The -F Command

Syntax: `-F` *filespec*

The ‘‘`-f`’’ command is effective only on the DOS command line. It instructs MS-Kermit to use the specified file as its initialization file, rather than `MSKERMIT.INI`. Unlike other command-line arguments, ‘‘`-f`’’ does not, of itself, cause MS-Kermit to exit upon completion. Example:

```
C>kermit -f sunday.ini
Kermit-MS>
```

The -F command line option allows different MS-Kermit initialization files to coexist. You can create batch commands to invoke Kermit in different ways, for instance `MONDAY.BAT` might contain ‘‘`kermit -f monday.ini`’’, `TUESDAY.BAT` ‘‘`kermit -f tuesday.ini`’’, etc.

The ECHO Command

Syntax: `ECHO [`*string*`]`

The ECHO command writes the string to the screen, without adding a carriage return or line feed. ECHO may be used to report progress during execution of a TAKE command file, or to issue prompts during the execution of a script.

```
ECHO Part one completed...\13
```

The number at the end is a "backslash codes" for ASCII control characters, in this case carriage return (`\13`). Since the ECHO command interprets backslash codes, `ANSI.SYS` and similar console drivers can be programmed through this command by embedding ANSI escape sequences (see section 1.17.3) in the echo string. The ECHO command always outputs a linefeed before the string.

The COMMENT Command

Syntax: `COMMENT` *text*

The COMMENT command lets you add comments to a TAKE command file. The word COMMENT (or any unique prefix thereof) must appear as the first word on the line. The COMMENT command may also be entered interactively. It has no effect at all. Example:

```
COMMENT - MS-Kermit command file to connect port 2 to an IBM mainframe
set port 2
set speed 4800   ; Transmission rate is 4800
do ibm           ; Set parameters for IBM linemode
connect          ; Be a terminal
```

Question marks can be included in comments without invoking the help function.

1.6.2. Local File Management Commands

These commands are executed on your local PC, and generally invoke DOS services. This allows you to perform common DOS functions without leaving Kermit. All file specifications may include device and/or directory fields. The local file management commands are:

`CWD` *path*
: Changes the current working directory to the given path. All references to local file names without explicit paths will refer to that path. A drive letter may be included to also change disk drives. This command affects Kermit and any inferior programs that you RUN or PUSH to, but your previous disk and directory are restored when you exit from Kermit. For consistency with DOS, you may also type CD.

`DELETE` *filespec*
: Deletes the specified file or files. As in DOS, the names of the deleted files are not listed, only the message "file(s) deleted" or "file(s) not found", and if you give the command "delete `*.*`", Kermit-MS will prompt "Are you sure?" since DOS is doing the work.

`DIRECTORY` [*filespec*]
: Lists the names, sizes, and creation dates of files that match the given file specification. If no filespec is given, the command is equivalent to `DIR *.*`. Normal DOS switches are effective.

`SPACE` Tells how much space is available on the current disk.

`RUN` *command*
: Passes the command line to `COMMAND.COM` for execution. Any legal DOS operation is permitted: running a program (perhaps with command line arguments or i/o redirection), executing a DOS command, or executing a batch file. Kermit is suspended while the command is executed and automatically resumes afterward. The command will be executed directly by `COMMAND.COM` so follow the rules of DOS. Example:

```
Kermit-MS>run more < xmas.txt
```

`TYPE` *filespec*
: Displays the specified local file on the screen. Automatic pause is not available at the end of a page (but see above example for how to accomplish this). On most systems, Ctrl-S can be typed to stop scrolling and Ctrl-Q to continue scrolling.

In most cases when you issue a local command, Kermit attempts to run the equivalent DOS command. If you get a message like "`?Unable to execute program`", it means that Kermit could not find `COMMAND.COM`, or that there was not enough memory left to load it. To ensure that Kermit can find `COMMAND.COM`, you should include a PATH statement in your `AUTOEXEC.BAT` file, which includes the device and directory where `COMMAND.COM` resides.

You can add your own local commands by defining macros for them. For example:

```
define edit run epsilon \%1
define more run more < \%1
define rename run ren \%1 \%2
```

Then you can use these commands at Kermit-MS prompt level: "`edit foo.bar`", "`more oofa.txt`", "`rename old.txt new.txt`". However, you cannot redefine built-in commands, for example:

```
define send receive \%1
```

See Section 1.7 for further information about macros.

1.6.3. COMMANDS FOR TERMINAL CONNECTION

The CONNECT command connects your PC as a terminal to the remote system so that you may conduct a session there, and the HANGUP command may be used to disconnect your modem (if you have one) from the remote system. There is presently no built-in DIAL command. Modems may be dialed "manually" during CONNECT, or you can construct your own DIAL command by using scripts, which are described in detail in subsequent sections.

For completeness, the descriptions below contain copious reference to the SET commands, which let you modify all sorts of terminal and communication parameters (the SET commands are described in a later section). MS-Kermit is initially set up with the following parameters, so that you only need to issue SET commands for those that need to be changed:

PORT	1 (in most cases, e.g. COM1 on the IBM PC family)
TERMINAL	VT102(*) emulation (IBM PC, DEC Rainbow)
SPEED	Whatever the serial card is currently set to.
PARITY	None
FLOW-CONTROL	XON/XOFF
HANDSHAKE	None
LOCAL-ECHO	Off
DISPLAY	7-bit characters
INPUT TRANSLATION	Off
ESCAPE	Control-Rightbracket

(*) The VT102 terminal is compatible with the VT100, but includes a few additional functions.

The CONNECT Command

Syntax: `CONNECT` *-or-* `C`

The CONNECT command establishes an interactive terminal connection to the remote system using the currently selected communications port (SET PORT COM1 or COM2, COM1 is the default) with all settings currently in effect for that port, emulating the currently selected type of terminal.

During CONNECT, the characters you type are sent out the communication port, and the characters that arrive at the port are displayed on the screen or interpreted by the selected terminal emulator. If you SET LOCAL-ECHO ON, MS-Kermit itself will display the characters you type on the screen.

Before you issue the CONNECT command, be sure to set the correct communication speed (SET SPEED) and any other necessary communication parameters (e.g. SET PARITY, SET LOCAL-ECHO). If you have SET DEBUG ON, then (on most DOS systems, particularly the IBM PC), received control characters will be displayed in special notation and no particular terminal will be emulated.

By default, 7-bit ASCII characters are displayed on the screen. If you SET DISPLAY 8, then 8-bit characters will be used (useful for "national" character sets). Character translation will be done according to any SET TRANSLATION INPUT and SET KEY commands you have issued. In addition, characters that are sent to the screen will also be recorded in a disk file or on a printer if you have issued a LOG SESSION command.

The CONNECT command turns your PC into a terminal to the other computer. To get back to the PC, type the escape character followed by the letter C (for "Close connection"). On most MS-DOS systems the escape character is Ctrl-`]` (Control-Rightbracket). That means, hold down the Ctrl key, press ‘‘`]`’’, and then type the letter C.

```
Kermit-MS>connect              Connect to remote system.

   Conduct terminal session here . . .

^]c                            Escape back to PC.
Kermit-MS>                     Prompt reappears.
```

This is called "escaping back". You can use the SET ESCAPE command to change the escape character to something besides ‘‘^]’’, or you can assign the escaping-back operation to a single key or key combination with SET KEY (on the IBM PC the default for this is Alt-X).

You can include the CONNECT command in a TAKE command file, but not "bare" text to be sent to the remote system during CONNECT (use scripts for that, see Section 1.8). When a TAKE file includes a CONNECT command, no further commands will be executed from the file until after you escape back. A curious side effect of allowing Kermit to accept input redirected from a file or device is that Connect mode will read characters from that file or device; not really that useful but it works if you happen to need it.

When you CONNECT, the program attempts to raise the DTR and RTS RS-232 signals (see Table 1-3), and it takes no specific action to lower them unless you explicitly issue the HANGUP command; thus you can EXIT from Kermit-MS and restart it without dropping a dialup connection. While CONNECTed, you can communicate directly with an autodialer or "smart modem" to control the communications line, hang it up, and the like, for instance, by typing AT commands to a Hayes-like modem.

```
Kermit-MS>set speed 2400        (See Section 1.6.10)
Kermit-MS>connect
AT                              Now you're talking to the modem.
OK                              Your modem responds
ATDT8765432                     Type the modem's dialing command.
RINGING
CONNECT 2400
Welcome to ...                  Now you're talking to the host computer.
Please login:
```

MS-Kermit makes no attempt to monitor the modem’s Carrier Detect (CD) or Data Set Ready (DSR) signals (see Table 1-3), and will take no notice if they drop. Thus it is not possible to automatically terminate a session if the connection is broken. However, you may query or test the status of these modem signals yourself using Kermit’s SHOW MODEM, SHOW COMMUNICATIONS, and WAIT commands.

Signal	DB25	DB9	Description
FG	1	-	Frame (protective) ground
TD	2	3	Transmitted data (from PC to modem)
RD	3	2	Received data (by PC from modem)
RTS	4	7	Request to Send (by PC)
CTS	5	8	Clear to Send (by modem)
DSR	6	6	Dataset Ready (Modem is turned on)
SG	7	5	Signal Ground
CD	8	1	Carrier Detect (Modem is communicating with remote modem)
DTR	20	4	Data Terminal Ready (PC is online)
RI	22	9	Ring Indicate (Modem tells PC phone is ringing)

Table 1-3: RS-232-C Modem Signals

When using Kermit to connect two PCs "back to back," SET LOCAL-ECHO ON so that when you CONNECT to the other PC to send messages to its operator, you can see what you are typing. You should also SET TERMINAL NEWLINE ON, so that that a linefeed will be automatically supplied for each carriage return you type.

The HANGUP Command

On serial port connections, the HANGUP command attempts to momentarily lower the modem signals DTR and RTS (Table 1-3). It may be used to hang up the phone when dialed up through a modem, or to get the attention of port contention units or terminal concentrators that operate in this manner. On direct connections, it will probably have no effect. On local area network connections, the network session is fully terminated. HANGUP affects only the currently selected port.

TERMINAL EMULATION

The IBM PC version of Kermit-MS emulates the DEC VT102 terminal by default, and may also be instructed to emulate the DEC VT52, the Heath/Zenith-19, the Tektronix 4010 graphics terminal, or no terminal at all, selectable with the SET TERMINAL command (or you may "toggle" among the different emulations by typing the Alt-Minus key). Emulation of each of these terminals is nearly complete. VT102 emulation lacks only smooth scroll and 132 column mode (132 column mode is supported for a number of popular EGA and VGA boards). Double-height, double-width characters are supported, but simulated using ordinary characters.

The IBM PC's 40-column (large character) screen mode may be used during CONNECT (but you may also have to inform the remote host that your screen width is 40). This can provide improved readability to visually impaired persons. To use 40-column mode, enter the DOS command "MODE 40" (or CO40 or BW40). Other screen sizes are also sensed and used automatically, provided you have set them from DOS, before starting Kermit.

On color monitors, the foreground and background colors may be set using SET TERMINAL COLOR, and inverse/normal video display may also be selected, along with many other terminal parameters. A complete list of the commands, default key configurations, and escape sequences accepted by the IBM PC Kermit terminal emulator is given in section 1.17.1. Non-IBM-compatible PCs have different terminal emulation options. See section 1.11.

Escape-Level Commands

The escape character, normally Control-`]`, is used to regain the attention of Kermit-MS during CONNECT (you can change the escape character using SET ESCAPE). When you type the escape character, Kermit-MS waits for you to follow it with a single character command. For instance, the single character command ‘‘`?`’’ produces a list of available single character commands. This command is executed immediately; it may not be edited, and the program does not wait for a carriage return to confirm it. Table 1-4 shows CONNECT escape-level commands available in Kermit-MS. Typing any other character (except the space bar, which is the "null command") after the

`?`	Help -- Lists the available single-character commands.
`0`	(the digit zero) Transmit a NUL (ASCII 0).
`B`	Transmit a BREAK signal.
`L`	Transmit a Long BREAK signal (on some systems).
`C`	Close the connection and return to Kermit-MS prompt level.
`H`	Hangup the phone by lowering DTR and CTS momentarily.
`F`	File the current screen in the screen dump file.
`M`	Toggle the mode line, i.e. turn it off if it is on or vice versa.
`P`	Push to DOS; get back to CONNECT by typing EXIT.
`Q`	Temporarily quit logging the remote session.
`R`	Resume logging the remote session.
`S`	Show the status of the connection.
`^]`	(or whatever you have set the escape character to be) Typing the escape character twice sends one copy of it to the connected host.

Table 1-4: Kermit-MS Single-Character CONNECT Escape Commands

escape character will cause Kermit-MS to beep, but will do no harm. These actions are also Kermit action verbs and can be assigned to single keys. See SET KEY for details.

The Mode Line

When you first issue the CONNECT command, a message (on some systems, an inverse video "mode line") will display the most important facts about the connection you've just established, so that you can quickly diagnose any problems. Here's what the IBM PC mode line looks like:

```
Esc-chr:^] help:^]? port:1 speed:9600 parity:odd echo:rem VT102 .... PRN
```

This shows that the escape character is Ctrl-Rightbracket, that you would type Ctrl-rightbracket followed by question mark (‘‘`^]?`’’) to get help during CONNECT, that you are connected on port 1 at 9600 baud with odd parity and remote echo, and that a VT102 terminal is being emulated. The four dots represent the VT102s LEDs (they turn into the digits 1,2,3,4 when "lit") and PRN will show up if the printer is activated (e.g. by Ctrl-PrintScreen).

The mode line may be turned on and off using SET MODE, or the CONNECT escape character followed by the letter M.

Screen Rollback

On the IBM PC and some other systems (see Table 1-7), Kermit-MS provides several pages of screen memory which let you recall earlier terminal screens. These may be scrolled up and down using keys as shown in Table 1-8. For instance, the IBM PC uses PgUp (previous screen), PgDn (next screen), Ctrl-PgUp and Ctrl-PgDn (one line at a time), Home (top of screen memory), and End (bottom of screen memory). Lines that scroll off the top of the screen are saved. When an application clears the screen using a recognized screen-clear sequence (ESC [2 J), the whole screen is saved. The screen scrolling functions may be assigned to different keys with the SET KEY command.

If you have rolled the screen back and a new character must be displayed, it will normally appear at the current cursor position on the old screen. This is useful when you are trying to copy something from a previous screen. If you wish new characters to appear in their proper place on the "newest" screen, you can SET TERMINAL ROLL ON.

The number of lines in the roll back buffer depends on the machine, 10 full screens for IBM PCs and DEC Rainbows, and on the amount of memory available in the machine. Each screen needs 4KB on IBM PCs. Denser displays receive fewer roll back lines.

Screen Dump

The screen dump feature writes the contents of the current screen to a file (`KERMIT.SCN` unless another file was selected by the SET DUMP command) when the CONNECT escape-level command F is typed. The screen dump file is appended to on each successive screen dump, with each screen separated by a formfeed (Ctrl-L). This feature may be used in conjunction with screen rollback -- a handy way to recapture screenfuls of laboriously typed-in text after a remote host has crashed without saving your work. The corresponding action verb is "dump". Screen dump does not function when in Tektronix graphics mode; instead one of many graphics screen capture programs may be used independently commonly via the DOS Shift PrtSc key combination or by LOGging the incoming byte stream.

A screen dump differs from a session log in two ways. First, each desired screen must be manually filed, and second, the screen dump file has been stripped of any escape sequences, whereas the session log records them (see LOG SESSION).

Printer Control

During terminal emulation, a locally attached printer may be controlled in the normal manner, on most systems. Pushing the "Print Screen" key (shifted on some systems) will cause the current contents of the screen to be printed by DOS; holding down Ctrl while depressing Print Screen will alternately start and stop the spooling of incoming characters to the printer. On the IBM PC, the mode line will show PRN when the printer is activated in this manner. `^P` or `^N` are sent to the host during terminal emulation and do not toggle printing as they do when you're talking directly to DOS. CTRL-Print-Screen can be simulated with the Kermit-MS LOG PRN and CLOSE commands. VT102 (ANSI) style host-controlled transparent printing is also supported on the IBM PC. See section 1.18.6 for technical information about MS-Kermit's printer control.

Unix users may use the following shell script to print files on a locally attached printer:

```
#!/bin/sh
# pcprint
# usage: pcprint file(s)
#  or <any UNIX process that writes to standard output> | pcprint
#
echo -n '<ESC>[5i'
if [ $# -eq 0 ]; then
  cat
else
  cat $*
fi
echo -n '<ESC>[4i'
```

Note that "`<ESC>`" above should be replaced by a real Escape, ASCII character 27.

Graphics

MS-Kermit on the IBM PC, compatibles, and several other systems, is capable of emulating a Tektronix 4010 graphics terminal, for use with host-based software that can generate Tektronix control codes. When you enter Tektronix emulation, your cursor will disappear. Don't be alarmed, this is how Tektronix terminals behave.

The Tektronix emulator implements a mixture of Tek 4010 and 4014 features to draw characters, lines, and dots in graphics mode. These Tektronix terminals have a graphics display 780 dots high by 1024 dots wide. They use storage tube technology whereby a dot stays illuminated until the full screen is erased. They also lack cursor keys. Kermit's Tek emulator maps the 1024 by 780 dot display to the PC's current screen dimensions, say 640 across by 200 or 350 dots high, and retains limited use of the cursor keys. It automatically senses the active display adapter (EGA, CGA, Hercules, Mono, and AT&T/Olivetti style 640x400) and retains screen coloring (EGA) and the current graphics image (EGA and Hercules) if the adapter has sufficient memory. Automatic sensing can be manually overriden to select a particular display mode, such as VGA (640x480), by SET TERMINAL GRAPHICS <display type>. Pure monochrome systems, of course, lack a graphics capability; in this case Kermit approximates the graphic image by writing dots as plus signs.

Tektronix graphics mode is entered two different ways, automatically and voluntarily:

1. Automatically (which you can prevent via the Kermit command DISABLE TEK). While emulating a VT102, VT52, or Heath-19, reception of the byte pair ESCAPE Control-L causes the PC to change to graphics mode, clear the screen, and obey new input as Tektronix commands. A second automatic entry is reception of the escape sequence ‘‘`ESC [ ? 3 8 h`’’ which does the same as above except the screen is not cleared. Automatic mode is exited by either reception of Control-X or ‘‘`ESC [ ? 3 8 l`’’ (lower case L), or by toggling the terminal type (ALT minus, Kermit verb`\KTermtype`) to VT102, or something other than TEK. (These ‘‘`ESC [ ? 3 8 h/l`’’ sequences derive from the DEC VT340 terminal.)

2. Voluntary mode is when terminal type TEK4010 is selected by the Kermit command SET TERMINAL TEK4010 or by toggling to it using Alt-Minus. It is exited by SET TERMINAL another-kind or by toggling to another kind. ENABLE or DISABLE TEK and the exit-Tek-mode escape sequences are not applicable to voluntary mode.

Here are several common questions about Tek mode, and their answers:

1. *"How do I escape from graphics mode back to being a regular terminal?"* Within CONNECT mode, you can type the `\KTermtype` key, which is assigned by default to Alt-Minus. Repeated pressing of this key "toggles" among Kermit's terminal types, VT102, VT52, Heath-19, and Tektronix. You can also escape back to Kermit-MS command level and issue an explicit SET TERMINAL command to change the terminal type.

2. *"How can I return to the graphics screen without erasing it?"* The graphics screen is preserved if your graphics adapter has sufficient memory (see Table 1-5). In this case, both your text and graphics screens will be preserved when you toggle back and forth between a character terminal (e.g. VT102) and Tektronix.

3. *"How do I erase the graphics screen?"* You can type the `\KReset` key, which is normally assigned to Alt-=. The screen also clears if the host sends a Control-L or ESC Control-L.

4. *"How do I print or save the graphics screen?"* Kermit does not currently provide a way to do this, but you can load drivers like `GRAPHICS.COM` alongside Kermit for this purpose.

While acting as a Tek terminal Kermit uses the keyboard translation appropriate to the VT102 terminal. However, received escape sequences are interpreted by the Tek emulator and VT102 escape codes are inoperative. The Tek emulator absorbs the ESCAPE and following character and treats any additional unknown items as ordinary text.

The emulator can display text characters from a built-in 8-by-8 dot font for characters Space through DELete (no control codes nor special characters). Tabs are converted to single spaces. Only the low 7 bits of the character are used.

While in Tek mode the emulator behaves as a simple TTY device for ordinary text and as a line or dot drawing Tektronix device for commands listed in Table 1-10. The screen resolution is governed by the kind of active display adapter and monitor in the PC (Table 1-5). Kermit senses this automatically when graphics mode is entered. Graphics are saved on page one of screen memory. Coloring is determined by the current terminal status, either the default screen or that overridden by the command SET TERMINAL COLOR.

Display Adapter	Display	Mode	Screen Resolution and Coloring
VGA	Hi res color	18	640x480, graphics saved (407 lines), 16 colors.
VGA	Monochrome	17	640x480, graphics saved (407 lines)
EGA w/256KB	Hi res color	16 dec	640x350, graphics saved, 16 colors.
	Med res color	14	640x200, graphics saved, 8 colors.
	Monochrome	15	640x350, graphics saved, b/w.
EGA w/64KB	Hi res color	16	640x350, graphics not saved, 4 colors of red, white, blue, black.
	Med res color	14	640x200, graphics saved, 8 colors.
	Monochrome	15	640x350, graphics not saved.
CGA	Color	6	640x200, graphics not saved, b/w.
Hercules	Monochrome	none	720x348, graphics saved if memory.
Monochrome	Monochrome	7	80 by 25 text, graphics not saved.
AT&T/Olivetti	any	72	640x400, grahics not saved, b/w.
DEC VAXMATE	any	208	640x400, graphics not saved, b/w.
TOSHIBA T3100	any	116	640x400, graphics not saved, b/w.

Table 1-5: Adapters Supported by IBM PC MS-Kermit for Tektronix Emulation

The technical details of Tektronix emulation are presented in section 1.17.7.

1.6.4. COMMANDS FOR FILE TRANSFER

MS-Kermit's SEND, GET, and RECEIVE invoke the Kermit file transfer protocol for error-checked transmission of files between MS-Kermit and another Kermit program on the other end of the connection. There are also commands for "raw" transfer of files (no error checking) with systems that don't have Kermit programs: LOG SESSION (for capturing text files on your PC) and TRANSMIT (for uploading text files to the remote system). The LOG TRANSACTION command opens a file to record the status, time, date, names, sizes of each file transfer.

During file transfer, MS-Kermit normally displays its progress on the screen as shown in Figure 1-1. The items in the right-hand column are updated more or less at random. The percent done is always filled in when sending files, and when receiving if the other Kermit sends the file's size in a special file-attribute packet. The number of retries indicates how many times Kermit had to correct transmission errors. Several other file transfer display format options are also available; see SET DISPLAY.

```
              Kermit-MS: V2.32  11 Dec 1988

              File name: FOT.
      KBytes transferred: 7
     Percent transferred: 52%
                Sending: In progress

      Number of packets: 74
          Packet length: 93
      Number of retries: 2
             Last error: None
           Last warning: None
```

Figure 1-1: MS-Kermit File Transfer Display Screen

Although MS-Kermit makes no distinction between text and binary files, most other Kermit programs do. Therefore, before you attempt to transfer binary files with another type of system (say, a VAX, or an IBM mainframe), be sure to give the appropriate command -- usually SET FILE TYPE BINARY -- to the Kermit on the remote end. Kermit-MS itself neither has nor needs the command SET FILE TYPE, because the MS-DOS format for text files is exactly the same as Kermit's text-file transfer format, which means that MS-Kermit never needs to convert file data, no matter whether it be text or binary.

File transfers involving floppy disks will be slow and noisy. Hard disks are much faster (and quieter), and RAM disks faster still (and totally silent). But if you store new files on a RAM disk, be sure to move them to a real disk before turning off your PC.

Before attempting to transfer files to the PC, make sure you have enough room on the selected device. Kermit does not provide a way for you to change disks during a file transfer. However, the Kermit protocol will help you out a little bit by attempting to prevent transfer of files that are too big to fit in the available space. As of version `2.31`, MS-Kermit supports "file attributes" exchange, and if the other Kermit supports this option too, then the receiving program will check free disk space before letting the transfer proceed. MS-Kermit allows a margin of 6 percent inflation upon reception, because file construction differs markedly between systems. A multiple-file transfer can even skip automatically past files that are too big, allowing the little ones to pass though.

Other attributes exchanged by MS-Kermit include the file's creation date and time, and the system of origin. When two Kermit programs both have attribute capability, then files will be stored with the same timestamp on the receiving system as they had on the sending system.

Since exchange of attributes is a new feature to MS-Kermit, and a relatively scarce one elsewhere, it is possible that two Kermit programs might misunderstand each other because of differing interpretations by the programmers, and this could prevent otherwise normal file transfers from taking place. An escape clause is provided by the command SET ATTRIBUTES OFF, which makes MS-Kermit forget that it has attribute capability.

You may record the progress of a file transfer in a log file by issuing the command LOG TRANSACTIONS.

The SEND Command

Syntax: `SEND` *filespec1* [*filespec2*]

The SEND command causes a file or file group to be sent from the local MS-DOS system to the Kermit on the remote system. The remote Kermit may be running in server or interactive mode; in the latter case, you should already have given it a RECEIVE command and escaped back to your PC. S is a special non-unique abbreviation for SEND.

filespec1 may contain the wildcard characters ''*'' to match zero or more characters within a field, and/or ''#'' (first position) or ''?'' (elsewhere) to match any single character (a question mark in first position gives you a help message). If *filespec1* contains wildcard characters then all matching files will be sent, in the same order that MS-DOS would show them in a directory listing. If *filespec1* specifies a single file, you may direct Kermit-MS to send that file with a different name, given in *filespec2*, as in:

```
Kermit-MS>send foo.bar framus.widget
```

filespec2 begins with the first nonblank character after *filespec1* and ends with the carriage return; thus it may contain blanks or other unusual characters that may be appropriate on the target machine. The alphabetic case of text in *filespec2* is preserved in transmission, so if case matters on the target system, be sure to type *filespec2* appropriately.

If the SEND command is specified by itself on the command line, then you will be prompted separately for the name of the file to send, and the name to send it under:

```
Kermit-MS>send
 Local Source File: c:\stuff\xcom1.txt
 Remote Destination File: com1.txt
```

If a file can't be opened for read access, the message "Unable to find file" will be shown or else the standard MS-DOS recovery procedures will take place:

```
Not ready error reading drive A
Abort, Retry, Ignore?
```

Kermit remains active even if you select "Abort" (DOS's word, not ours).

Files will be sent with their MS-DOS filename and filetype (for instance `FOO.TXT`, no device or pathname). Special characters in the file name are not converted. If there is no filetype, then only the name will be sent, without the terminating dot. Each file is sent as is, with no conversions done on the data, except for possibly stopping at a terminating Control-Z character (see the SET EOF command).

Once you give Kermit-MS the SEND command, the name of each file will be displayed on your screen as the transfer begins. Packet, retry, and other counts will be displayed along with informational messages during the transfer, in the style specified by SET DISPLAY. If the file is successfully transferred, you will see ''`Complete`'', otherwise there will be an error message. When the specified operation is done, the program will sound a beep.

Several single-character commands may be given while a file transfer is in progress:

`^X` (Control-X) Stop sending the current file and go on to the next one, if any.

`^Z` Stop sending this file, and don't send any further files.

`^C` Return to Kermit-MS command level immediately without sending any kind of notification to the remote system. (^Z or even ^E is preferable.)

^E Like ^C, but send an Error packet to the remote Kermit in an attempt to bring it back to server or interactive command level.

CR Simulate a timeout: resend the current packet, or NAK the expected one.

Control-X, Control-Z, and Control-E send the proper protocol messages to the remote Kermit to bring it gracefully to the desired state. Control-C leaves the remote Kermit in whatever state it happens to be in, possibly retransmitting its last packet over and over, up to its retry limit. You should only have to use Control-C in dire emergencies (the remote Kermit is stuck, the remote system crashed, etc), or at those times when you realize that you have given a file transfer command to Kermit-MS without first having told the remote Kermit about it.

MS-Kermit does not have a built-in mechanism for sending an entire directory structure, but this may still be done using command files. A program called XSEND, distributed along with MS-Kermit, will construct such a command file automatically.

The RECEIVE Command

Syntax: RECEIVE [*filespec*]

The RECEIVE command tells Kermit-MS to receive a file or file group from the other system. The file is stored under the name it was transmitted with, except that any illegal characters are translated to X's. Kermit-MS passively waits for the file to arrive; this command is not to be used when talking to a Kermit server (use GET for that). You should already have issued a SEND command to the remote Kermit and escaped back to Kermit-MS before issuing the RECEIVE command. The RECEIVE command is intended for situations where the file name and sending operation originates at the other side; GET originates the request from our side and asks the server to perform the operation. R is a special non-unique abbreviation for RECEIVE.

If the optional filespec is provided, incoming files will be stored under that name. If the filespec is really just a path then files are stored where the path indicates. If it is an actual filename the first incoming file is renamed and any additional files either overwrite the first (if FILE WARNING is OFF) or are renamed slightly from the filespec (digits are added to the end of the main filename part before the dot and extension) if FILE WARNING is ON (the default). The filespec may include any combination of the following fields:

Device designator
Store the file on the designated device, in the current directory for that device. If no device designator is given, store it on the current default device.

Directory path
Store the file in the designated directory on the current disk. If no path given, store the file in the current directory.

File name
Store the file under the name given. If no name is given, store it under the name it was sent under, converted, if necessary, to suit DOS conventions, and modified, if SET WARNING ON, to avoid overwriting any file of the same name in the same directory.

If an incoming file does not arrive in its entirety, Kermit-MS will normally discard it and it will not appear in your directory. You may change this behavior by using the command SET INCOMPLETE KEEP, which will cause as much of the file as arrived to be saved on the disk.

The same single-character commands are available as during SEND:

^X Request that the remote Kermit stop sending the current file, and proceed to the next one immediately. Since this is an optional feature of the Kermit protocol, the remote Kermit might not honor the request.

^Z Request that the remote Kermit terminate the entire transfer; this is also an optional feature that may or may not be supported by the remote Kermit.

^C, ^E, and CR operate in the same way as they do during SEND. In this case, ^E should always do what ^Z is supposed to do.

If WARNING is OFF and you type `^X` or `^Z` to interrupt the transfer, you'll either get a partial new file, or else both the old and the new file of that name will be lost, depending on SET INCOMPLETE. In any case, when WARNING is off, old files with the same name as incoming files will not survive.

Caution: If an incoming file's name (the part before the dot) corresponds to an MS-DOS device name, such as NUL, COM1, CON, AUX, or PRN, output will go to that device, rather than to a file with that name. This is a feature of MS-DOS.

1.6.5. Hints for Transferring Large Files

During a prolonged file transfer session, things can go wrong that are beyond Kermit's control. The longer the session, the greater the probability it will be fatally interrupted. But you can take a few precautions:

- Make sure there is sufficient disk space at the receiving end. If possible, first run a disk utility (such as CHKDSK) to clean out any bad disk blocks.
- If you are using a telephone connection, make sure your session won't be interrupted by call waiting, people picking up other extensions, etc.
- Don't attempt to transfer a single file of many megabytes over a telephone connection. The longer the call, the greater the chance of disconnection (carrier loss). Although it's a bother, it may save time in the long run to break the file up into smaller pieces, transfer the pieces, and then recombine on the other end.
- SET INCOMPLETE KEEP on the receiving end, so that if the transfer fails, then the partial file will be retained. Then chop the part that wasn't transferred into a separate file, reconnect, and send it. Then join the pieces together.

Consider moving truly massive amounts of data on magnetic media. "Never underestimate the bandwidth of a station wagon full of magnetic tapes!" (or diskettes).

1.6.6. Commands for Raw Uploading and Downloading

MS-Kermit can be used to send files to, or capture files from, remote systems that do not have Kermit programs available. No error checking or correction is done, so the results can very likely contain corrupted characters, spurts of noise, gaps, or extraneous system messages or prompts. The command for uploading is TRANSMIT, and for downloading LOG SESSION.

To minimize loss of data during these operations, be sure to SET the FLOW-CONTROL and HANDSHAKE parameters to match the characteristics of the system on the other end.

The TRANSMIT Command

Syntax: `TRANSMIT` *filespec* [*prompt-character*]

The TRANSMIT command provides a basic raw upload (export) facility to send straight ASCII text files to the host without packets, error checking, or retransmissions, but using all the currently selected communication parameters for flow control, parity, etc. Information is read from the disk file a line at a time, sent out the serial port, and the command waits for a single character prompt (normally linefeed) from the host before sending the next file line. A disk file line ends with carriage-return-linefeed (CRLF), but only the carriage return is sent, just as you only type carriage return at the end of a line, not CR and LF. Most remote systems will echo the CR and then also supply a LF, which indicates that they have processed the line and are ready for another one. Setting the prompt to binary zero, `\0`, makes the TRANSMIT command proceed without waiting for a prompt. Pressing the local Return key simulates arrival of a prompt character.

Typically, before using this command to upload a file, you would start a text editor (preferably a line-oriented, rather than full-screen, editor) on the remote host and put it into text insertion mode. When the file has been completely transmitted, you would manually enter the required sequence for getting the editor out of text insertion mode, and then make any necessary corrections by hand. Here's an example for VAX/VMS:

```
Kermit-MS>set flow xon/xoff         Set flow control to match VAX/VMS.
Kermit-MS>connect                   Connect to VAX.
$ edt foo.txt                       Start the EDT editor.
*i                                  Put it into "insert" mode.
^]c                                 Escape back to Kermit-MS.
Kermit-MS>transmit foo.txt          Upload the file a line at a time.
 ...                                The lines are displayed on your screen.
Kermit-MS>connect                   When done, connect back to the VAX.
^Z                                  Type Ctrl-Z to exit EDT insert mode.
*exit                               Exit from EDT to save the file.
$
```

If transmission appears to be stuck, you can wake it up by typing a carriage return on the keyboard. You can cancel the TRANSMIT command by typing a Control-C. Control-Z's or other control characters in the file may have adverse effects on the host. For this reason, you should use TRANSMIT only for files that contain 7-bit printing ASCII characters, spaces, tabs, carriage returns, linefeeds, and possibly formfeeds.

The LOG SESSION Command

Syntax: `LOG SESSION` [*filespec*]

The LOG SESSION command lets you copy the characters that appear on your screen during CONNECT into the specified file on the PC. You can use this command to download files by displaying (usually with a command like TYPE) the file on the remote system while logging is in effect. Example:

```
Kermit-MS>set flow xon/xoff         Set flow control to match VAX/VMS.
Kermit-MS>connect                   Connect to the VAX.
$ type foo.bar                      Give this command, but don't type CR yet.
^]c                                 Escape back.
Kermit-MS>log session foo.bar       Start logging.
Kermit-MS>connect                   Connect back.
                                    Now type the carriage return.
This is the file FOO.BAR.           The file is displayed on your screen
Blah blah ...                       and captured into PC file FOO.BAR.
$                                   The prompt is captured too.
^]c                                 When done, escape back
Kermit-MS>close session             and close the log file.
```

The PC file `FOO.BAR` now contains a (possibly mutilated) copy of the remote computer's `FOO.BAR` file. It probably has the remote system's prompt at the end, which you can edit out. The session log can also be used to record typescripts, editing sessions, Tektronix graphics output, or any other output from, or dialog with, the remote computer.

During terminal emulation, the LOG command records all the characters that arrive from the remote host in the specified file, including escape sequences, with any input character translations applied according to SET TRANSLATION INPUT. If you have SET LOCAL-ECHO ON, the characters you type will also be recorded. Logging may be suspended and resumed within a terminal session with the CONNECT escape-level commands Q and R. The log file will be composed of 7-bit ASCII bytes if (a) PARITY is other than NONE, or (b) DISPLAY is SET to 7. If DISPLAY is 8 and PARITY is NONE, or if DEBUG is ON, then the log will contain 8-bit bytes.

You may LOG SESSION PRN to cause the logging information to be printed directly on your printer. Any escape sequences that are sent to the screen are also sent to the printer.

If you want to record information without imbedded escape sequences, use the screen dump feature, invoked by the

CONNECT escape-level command F, which is described under the CONNECT command.

A session log cannot be played back directly on the PC from the log file. To relive the session, you must transfer it to the remote system and display it in "binary mode" (e.g. cat in Unix) while CONNECTed.

1.6.7. Kermit Server Commands

Kermit-MS can act as a Kermit server, and can also interact with other Kermit servers. Normally, the remote Kermit is put into server mode. Then the local Kermit becomes a "client", and may issue repeated commands to the server without having to connect and escape back repeatedly. Servers can not only transfer files, but can also provide a variety of file management functions. The SERVER command puts MS-Kermit into server mode, and the DISABLE and ENABLE commands modify the behavior of the server.

Kermit servers respond only to information sent as Kermit protocol packets and not to ordinary CONNECT-mode commands. When MS-Kermit is the client, it uses the SEND command (described above) to send files to a server, the GET command (*not* RECEIVE) to get files from a server, the REMOTE commands to invoke the file management functions of the server, and the BYE, FINISH, or LOGOUT commands to shut down the server. The MS-Kermit server can also be returned to interactive mode by typing Ctrl-C or Ctrl-Break on the PC's console keyboard; if the SERVER command was issued from a command file, execution of the command file will resume with the next command after SERVER.

The SERVER Command

Syntax: SERVER [timeout]

Kermit-MS is capable of acting as a full-fledged Kermit server for users coming in through one of the communication ports or a local area network. To put Kermit-MS into server mode, first issue any desired SET commands to select and configure the desired port, then DISABLE any undesired functions, and then type the SERVER command. Kermit-MS will await all further instructions from the client Kermit on the other end of the connection, which may be hardwired, or connected through a network or autoanswer modem.

In the following example, a Kermit server is set up for dialing in:

```
Kermit-MS>set port 1
Kermit-MS>set speed 1200
Kermit-MS>hangup
Kermit-MS>connect
ATS0=1
OK
^]c
Kermit-MS>set server timeout 0
Kermit-MS>set warning on
Kermit-MS>disable all
Kermit-MS>server
```

Before putting Kermit in server mode in this case it was necessary to connect to the modem (in this example, a Hayes) and put it into autoanswer mode by typing the ATS0=1 command. Since Kermit packets typically start with a Control-A character check the modem's manual to ensure that character is not a modem command signal; some brands regard Control-A as a hangup request!

Note the command SET SERVER TIMEOUT 0. This disables the MS-Kermit server's normal behavior of timing out periodically and sending a NAK packet while waiting for a connection. This might be necessary with certain modems or PBXs that can be taken out of answer mode if they receive any characters from the PC before a call is received.

An optional timeout value can be specified to exit server mode automatically at a certain time. The timeout can be expressed as a number, meaning seconds from now, or as the `hh:mm:ss` form, in 24-hour time of day. Both forms recognize times greater than 12 hours from now as being in the past. For instance, if you want to run a Kermit server for an hour, and then have it exit so that another program can run, use a command file like:

```
set port 1        ; Use COM1
set speed 2400    ; at 2400 bps.
disable all       ; Only allow file transfers in current directory.
server 3600       ; Be a server for 3600 seconds = 1 hour.
exit              ; Exit when done.
```

MS-Kermit `2.32` server mode supports the following requests:

SEND	REMOTE CWD (CD)	REMOTE MESSAGE
GET	REMOTE DELETE	REMOTE SEND
FINISH	REMOTE DIRECTORY	REMOTE SPACE
BYE	REMOTE HELP	REMOTE TYPE
LOGOUT	REMOTE HOST	REMOTE WHO
	REMOTE LOGIN	

REMOTE CWD (CD) can be used to change both directories and devices. The REMOTE MESSAGE command accepts a one line message on the command line which will be displayed on the operator's console. An MS-Kermit Server can DISABLE recognition of selected REMOTE commands to help reduce accidents.

> *CAUTION:* The method used for most of the REMOTE commands is to invoke a task with the user's command line, redirect standard output to a temporary file, `$KERMIT$.TMP`, send that file back to the remote end, and then delete the file. Sufficient space must be available to store this file. To service DOS commands or user tasks `COMMAND.COM` must be located on the DOS PATH.
>
> *FURTHER CAUTION:* Any of these DOS tasks or programs may encounter an error, and in that case, DOS will generally put the familiar "Abort, Retry, Ignore?" message on the server's screen, and will wait for an answer from the keyboard. This will hang the server until a human comes to the keyboard and gives a response. The same thing will happen when any program is invoked that interacts with the real console. DISABLE ALL seems to avoid most unpleasant situations of this kind.

For local network operation with NetBios, the SET PORT NET command (with no node name) must be issued before the SERVER command. MS-Kermit then becomes a network-wide server, and other client Kermits can start a network session with it by using the name of the Kermit Server, which is shown on the server's screen when SET PORT NET is given. The Kermit Server accepts connections from other Kermits, but only one at a time. There may be many Kermit Servers active on the network simultaneously because each has a unique node name. Operations are exactly the same as with serial port usage and the session (equivalent to a dialed phone connection) is maintained between the pair until too many timeouts occur, or the client Kermit issues a HANGUP command, exits to DOS, or SETs PORT NET to another node. In the latter cases, the server remains available for use by other client Kermits. If a client Kermit issues the BYE or FINISH command, the network server is shut down (unless it was started with FIN disabled).

The DISABLE and ENABLE Commands

For security purposes, it may be desirable to leave your PC in Kermit server mode so that it can be dialed in to, but with certain functions unavailable to those who dial in. The DISABLE and ENABLE commands provide this control.

The DISABLE and ENABLE commands affect the following functions, with the effect of DISABLEs noted:

CWD	(CD) Changing of directories, disabled entirely.
DEL	Deletion of files confined to current directory.
DIR	Production of directory listings confined to current directory.
FIN	Shutting down the server (applies also to BYE) disabled entirely.
GET	Getting files from the server confined to current directory.
HOST	Execution of all REMOTE HOST (DOS) commands disabled entirely.
SEND	Forces files sent to server into current directory.
SPACE	Asking the server for a disk space report, disabled.

TYPE	REMOTE TYPE files confined to current directory.
ALL	All of the above.
TEK	Automatic invocation of Tektronix graphics mode by host commands. This function is not related to server mode, and is not included in the ALL term.

For reasons which should be obvious, the Kermit server does not provide a REMOTE ENABLE command!

The GET Command

Syntax: `GET` *remote-filespec*

The GET command requests a Kermit server to send the file or file group specified by *remote-filespec*. This command can be used only when Kermit-MS has a Kermit server active on the other end of the connection. This usually means that you have CONNECTed to the other system, logged in, run Kermit there, issued the SERVER command, and escaped back (e.g. ‘‘`^]C`’’) to the local Kermit-MS. In the case of LAN operation, a Kermit server must be running somewhere on the network. If the remote Kermit does not have a SERVER command, then you should use SEND and RECEIVE as described above.

You may use the GET command in a special way to specify a different name for storing the incoming file. Just type GET alone on a line, and you will be prompted separately for the remote filespec and the local filespec:

```
Kermit-MS>get
 Remote Source File: com1 txt
 Local Destination File: a:xcom1.txt
```

The local file name may contain a device field, and/or a directory specification. Device and directory specifications in the local destination file name work the same way as in the RECEIVE command. The multiline GET command is provided so that the distinction between the two files is always clear, which would not otherwise be the case if the foreign filename had spaces in it.

The remote filespec is any string that can be a legal file specification for the remote system; it is not parsed or validated locally. It can contain whatever wildcard or file-group notation is valid on the remote system, including spaces. If the string needs to begin with a question mark (?) then use a sharp sign (#) instead to avoid Kermit's help message; it will be transmitted as a question mark.

Once the file transfer begins, the GET command behaves exactly like the RECEIVE command.

Warning: If the remote filespec is to contain a semicolon, *and* the GET command is being issued from a TAKE command file, you must prefix the semicolon with a backslash. Otherwise, all characters beginning with the semicolon will be ignored:

```
get me.home\;2
```

1.6.8. Commands for Controlling Remote Kermit Servers

The BYE, FINISH, and LOGOUT commands allow you to shut down a remote Kermit server:

`BYE` When communicating with a remote Kermit server, use the BYE command to shut down the server, log out its job, and exit locally from Kermit-MS to DOS. On local area networks, BYE also terminates the network session.

`FINISH` Like BYE, FINISH shuts down the remote server. However, FINISH does not log out the server's job. You are left at Kermit-MS prompt level so that you can connect back to the job on the remote system. On local area nets, FINISH shuts down the MS-Kermit server, but in a way that allows it to be restarted as if no interruption had occurred.

`LOGOUT` The LOGOUT command is identical to the BYE command, except you will remain at Kermit-MS prompt level, rather than exit to DOS, so that you can establish or use another connection without having to restart MS-Kermit.

The REMOTE Commands

The REMOTE keyword is a prefix for a number of commands. It indicates that the command is to be performed by a remote Kermit server. Not all Kermit servers are capable of executing all of these commands, and some Kermit servers may be able to perform functions for which Kermit-MS does not yet have the corresponding commands. In case you send a command the server cannot execute, it will send back a message stating that the command is unknown to it. If the remote server can execute the command, it will send the results, if any, to your screen.

Here are the REMOTE commands that Kermit-MS may issue:

`REMOTE CWD` [*directory*]
: (Also REMOTE CD) Ask the server to Change your Working Directory on the remote host, that is, the default source and destination area for file transfer and management. You will be prompted for a password, which will not echo as you type it. If you do not supply a password (i.e. you type only a carriage return), the server will attempt to access the specified directory without a password. If you do not supply a directory name, your default or login directory on the remote system will be assumed and you will not be prompted for a password.

`REMOTE DELETE` *filespec*
: Ask the server to delete the specified file or files on the remote system. In response, the server may display a list of the files that were or were not successfully deleted.

`REMOTE DIRECTORY` [*filespec*]
: Ask the server to display a directory listing of the specified files. If no files are specified, then the list should include all files in the current working directory.

`REMOTE HELP`
: Ask the server to list the services it provides.

`REMOTE HOST` [*command*]
: Ask the server to send the command to the remote system's command processor for execution.

`REMOTE KERMIT` *command*
: Send the command to the remote Kermit for interpretation as a Kermit command in the remote Kermit server's own command syntax.

`REMOTE LOGIN` *user*
: Password and account are always solicted via prompts. A carriage return response corresponds to an empty entry. REMOTE LOGIN applies only to a remote Kermit server and not to a remote operating system; an MS Kermit server does not understand the command.

`REMOTE MESSAGE` *text*
: Send the one line text message to be displayed on the Server's screen.

`REMOTE SPACE` [*directory*]
: Ask the server to provide a brief summary of disk usage in the specified area on the remote host or, if none specified, the default or current area.

`REMOTE TYPE` *filespec*
: Ask the server to display the contents of the specified remote file or files on your screen.

`REMOTE WHO [`*who-spec*`]`
: Ask the server to list actively logged on users; optional who-spec qualifies the list and uses the syntax of the server system.

The Mail Command

Syntax: `MAIL` *filespec address*

The MAIL command is a very close relative of Kermit's SEND command. Mail sends a file, or file group, to a Kermit server with instructions (in an Attribute packet) to submit the file(s) to the host's Mailer utility rather than store them on disk. To round out a mail request a field following the filename is required, and into it we place the address to which the files are to be mailed. Mail addresses vary substantially, but several common forms are "username", "username@host", and "host`::`username". The MAIL command will work only if the Kermit server understands it, otherwise the mail request will be rejected before any files are sent. Kermit-MS can send mail but it cannot receive it, because MS-DOS does not have a mail facility. When sending, there is no way to transmit any fields other than the recipient's address and the message body; fields like subject and cc are not supported.

1.6.9. The LOG and CLOSE Commands

Syntax: `LOG {PACKET, SESSION, TRANSACTION}` [*filespec*]
`CLOSE {PACKET, SESSION, TRANSACTION}`

The LOG command tells MS-Kermit to record the terminal session, file transfer transactions, or the file transfer protocol packets themselves in a log file. If the log file already exists then new material is appended to it. Open log files may be closed (and the associated logging disabled) using the CLOSE command. Open log files are also closed when you EXIT from Kermit.

LOG SESSION is used to record your terminal emulation typescript. It was described above, in the section on file transfer.

The LOG TRANSACTION Command

Syntax: `LOG TRANSACTION` [*filespec*]

The Transaction log is a file recording a pair of text lines describing each file transfer (SEND, GET, RECEIVE, or some REMOTE commands). The lines indicate the local filename (and remote name if different), the time and date of the start of the transfer, the number of bytes transferred, and the status of the transfer. New entries are always appended to old to prevent loss of records. The default filename is `TRANSACT.LOG`. The command SHOW LOGGING displays the current names and which logs are active. The command CLOSE TRANSACTION will voluntarily terminate this class of log; otherwise, it will be closed automatically when Kermit exits.

The LOG PACKETS Command

Syntax: `LOG PACKETS` [*filespec*]

The packet log is for diagnostic purposes and records each Kermit protocol packet sent and received in printable format. Control characters are written as caret-letter and characters with the high bit set are shown as their 7-bit part preceeded by a tilde. The default filename is `PACKET.LOG`. If you experience difficulty with file transfers the packet log is valuable in discovering who said what to whom, even though a copy of the Kermit book is needed to unravel the meaning of each character in a packet.

1.6.10. The SET Command

Syntax: `SET` *parameter* [*parameter*] *value*

The SET command establishes or modifies parameters for file transfer or terminal connection. You can examine their values with the SHOW or STATUS commands. The following SET commands are available in Kermit-MS:

ALARM	Set alarm clock time, for IF ALARM testing
ATTRIBUTES	Controls whether MS-Kermit uses Attribute packets
BAUD	Communications port line speed (synonym for SPEED)
BELL	Whether to beep at the end of a transaction
BLOCK-CHECK-TYPE	Level of error checking for file transfer
COUNT	Variable for TAKE file and macro IF COUNT testing
DEBUG	Display packet contents during file transfer
DEFAULT-DISK	Default disk drive for file i/o
DELAY	Wait number seconds before Sending a file
DESTINATION	Default destination device for incoming files
DISPLAY	For selecting the type of file transfer display
DUMP	Screen dump file (or device) name
END-OF-LINE	Packet termination character
EOF	Method for determining or marking end of file
ERRORLEVEL	Value returned to DOS Batch files
ESCAPE	Escape character for CONNECT
FLOW-CONTROL	Enable or disable XON/XOFF
HANDSHAKE	Half-duplex line turnaround option
INCOMPLETE	What to do with an incompletely received file
INPUT	Behavior of INPUT command for scripts
KEY	Specify key redefinitions
LOCAL-ECHO	Specify which computer does the echoing during CONNECT
MODE-LINE	Whether to display a mode line during terminal emulation
PARITY	Character parity to use
PORT	Select a communications port
PROMPT	Change the "`Kermit-MS>`" prompt to something else
RECEIVE	Request remote Kermit to use specified parameters
REMOTE	For running Kermit-MS interactively from back port
RETRY	Packet retransmission threshold
SEND	Use the specified parameters during file transfer
SERVER	Parameters for server mode (command wait timeout)
SPEED	Communications port line speed (synonym for BAUD)
TAKE-ECHO	Control echoing of commands from TAKE files
TERMINAL	Emulation and parameters
TIMER	Enable/disable timeouts during file transfer
TRANSLATION	Enable/disable/specify conversion of arriving characters
WARNING	Specify how to handle filename collisions

The SET commands are now described in detail, in alphabetical order.

SET ALARM

Syntax: `SET ALARM` {*seconds, hh:mm:ss*}

The alarm is a timer, like an alarm clock, available for testing by IF ALARM statements. The alarm time is given as seconds from the present or as a 24-hour specific time of day. Both need to be within 12 hours of the present to avoid being mistaken for times in the past. SHOW SCRIPT displays the current alarm setting.

SET ATTRIBUTES

Syntax: `SET ATTRIBUTES {ON, OFF}`

Disables or enables use of Kermit file Attribute protocol packets, which contain the size, time, and date of files transferred using the Kermit protocol. This command is a safety feature so that a small misunderstanding with another Kermit cannot block transfers. SHOW FILE tells whether attributes are on or off; they are normally ON.

SET BAUD

Syntax: `SET BAUD` *number*

Synonym for SET SPEED (q.v.).

SET BELL

Syntax: `SET BELL {ON, OFF}`

Specifies whether the bell (beeper) should sound upon completion of a file transfer operation. Normally ON.

SET BLOCK-CHECK-TYPE

Syntax: `SET BLOCK-CHECK-TYPE {1, 2, 3}`

Selects the error detection method: a 1-character 6-bit checksum (the normal case), a 2-character 12-bit checksum, or a 3-character 16-bit cyclic redundancy check (CRC). If the other Kermit program is not capable of type 2 or 3 checking methods, automatic fallback to type 1 will occur. The more secure type 2 and 3 block checks take essentially no more execution time than the simple 1 character checksum. SET BLOCK 3 is a stronger check than SET BLOCK 2. SET BLOCK 2 or 3 is recommended for use with long packets (see below), noisy communication lines, binary (8-bit data) files, and text files containing critical data (budgets, grades, etc).

SET COUNT

Syntax: `SET COUNT` *number*

Set the value of the script COUNT variable to be between 0 and 65535. COUNT is used with IF COUNT to construct counted loops in script TAKE files and macros. Each active TAKE file or macro uses a private version of COUNT. The default value is zero, and the SHOW SCRIPT command displays the current value (meaningful only when given within a TAKE file or macro).

SET DEBUG

Syntax: `SET DEBUG {PACKET, SESSION, ON, OFF}`

With DEBUG PACKET, Kermit will display the actual packets on your screen during file transfer. With the normal file transfer display, regular-length packets sent and received are displayed in fixed-size slots. The display of extended-length packets, however (see SET RECEIVE PACKET-LENGTH), tends to overlap. If this bothers you, then also SET DISPLAY SERIAL, or LOG the packets rather than displaying them.

With DEBUG SESSION, during terminal emulation (on the IBM PC, Rainbow, and a few others), control characters are displayed in uparrow (‘‘^’’) notation and characters with the 8th bit set are preceded by the tilde (‘‘~’’) sign, and your session log (if any) will record 8-bit bytes, rather than 7-bit ASCII, regardless of SET DISPLAY or SET PARITY. Character translation (SET TRANSLATION INPUT) is not done during session debugging. The effect of SET DEBUG SESSION during terminal connection can be disconcerting, but it gives you a convenient line monitor equivalent to a specialized device that costs several thousand dollars, and it can prove very handy for tracking down data communication problems.

SET DEBUG ON turns on both SESSION and PACKET debugging, and SET DEBUG OFF turns them both off.

SET DEFAULT-DISK

Syntax: `SET DEFAULT-DISK` *x*`:[`*directory*`]`

Specify the default disk drive to use for file transfer, directory listings, and so forth. Equivalent to typing the DOS command for changing disks (`A:`, `B:`, etc). Affects Kermit and all inferior processes, but when you exit from Kermit, you will still have the same default disk as when you entered. As a convenience, a directory may be specified with or without the drive to change one or the other or both. This command is a synonym for CWD (CD).

SET DELAY

Syntax: `SET DELAY` *number*

Wait the specified number of seconds before starting a file transfer. Intended for use when the other side needs appreciable time to become ready, such as rearranging cables, changing programs, etc., or when MS-DOS Kermit is the remote Kermit (e.g. after CTTY COM1, SET REMOTE ON). The *number* is 0 to 63 seconds, normally 0.

SET DESTINATION

Syntax: `SET DESTINATION {DISK, PRINTER, SCREEN}`

SET DESTINATION PRINTER will cause incoming files to be sent directly to the printer; SCREEN will send output normally destined for the disk to the screen. The normal destination is DISK. SET DESTINATION affects only files transferred with SEND, GET, or RECEIVE; it cannot be used to reroute the output from REMOTE server commands.

SET DISPLAY

Syntax: `SET DISPLAY {QUIET, REGULAR, SERIAL, 7-BIT, 8-BIT}`

During file transfer, MS-DOS Kermit's regular display is a formatted screen whose fields are randomly updated with file names, packet numbers, error counts, percent done, error messages, and so forth, as shown in Figure 1-1.

If you wish to run Kermit-MS interactively through the back port, for instance after the operator has done CTTY COM1, you must give the command SET REMOTE ON (which, currently at least, is equivalent to SET DISPLAY QUIET); this suppresses the file transfer display screen, so that the display won't interfere with the file transfer itself. You can also use this command to suppress the display in local mode, in case you are using a system that allows you to do other work while file transfer proceeds in the background.

If you have your PC connected to a speaking device (a common practice for visually impaired people), or you are logging the display screen to a printer (using DOS `^P` or `kermit > prn`), the random nature of the regular display will make the results of little use. SET DISPLAY SERIAL is provided for this purpose; it causes the program to report progress "serially" on the screen. In serial mode, error messages are preceeded with the word "Error" and repeat messages with the word "Retry". Packets are numbered as dots with every tenth being a plus sign. The packet display is automatically broken across lines at every 70th packet. The serial display makes much more sense when spoken than does the regular display.

The serial display does not show the percent and kilobytes transferred. It is the default display style for generic MS-DOS Kermit; REGULAR is the default for all others.

The last two parameters, 7-BIT and 8-BIT, control the size of characters sent to the screen during terminal emulation. 7-BIT is the default and includes all ASCII characters. 8-BIT is useful with national and line drawing characters.

SET DUMP

Syntax: `SET DUMP` *filespec*

On those systems that support this feature, change the file or device name of the screen dump file. The normal file name is `KERMIT.SCN`. See the section on terminal emulation for details about screen dumps. If the specified file already exists then new material is appended to old. If you want to start a new screen dump file, delete the old one first.

SET END-OF-LINE

Syntax: `SET END-OF-LINE` *number*

If the remote system needs packets to be terminated by anything other than carriage return, specify the decimal value, 0-31, of the desired ASCII character. Equivalent to SET SEND END-OF-LINE (SET END-OF-LINE is kept only for historical reasons, and the parameter really should be called END-OF-PACKET anyway.)

SET EOF

Syntax: `SET EOF {CTRL-Z, NOCTRL-Z}`

Controls how the end of file is handled. CTRL-Z specifies a Control-Z character should be appended to the end of an incoming file. Certain MS-DOS text editors and other applications require files to be in this format. For outbound files, treat the first Control-Z as the end of the local file, and do not send it or any subsequent characters. NOCTRL-Z is the default; incoming files are stored, and MS-DOS files are sent, exactly as is, in their entirety. Use SHOW FILE to see the current SET EOF status.

SET ERRORLEVEL

Syntax: `SET ERRORLEVEL` *number*

Forces the DOS "errorlevel" variable to a given value. This is used in scripts when other controls or tests determine that the cumulative errorlevel reported to DOS Batch when Kermit exits needs to be modified. The number can be 0 to 255 decimal.

SET ESCAPE

Syntax: `SET ESCAPE` *character*

Specify the control character you want to use to "escape" from remote connections back to Kermit-MS. On most systems the default is ‘‘`^]`’’ (Control-Rightbracket), which was chosen because it is a character you would otherwise rarely type.

The *character* is entered literally after SET ESCAPE or in backslash number form (`\29`), and should be chosen from the ASCII control range. It is not possible to use non-ASCII characters (like function keys) for this purpose (but see SET KEY for a way around this restriction).

SET FLOW-CONTROL

Syntax: `SET FLOW-CONTROL {XON/XOFF, NONE}`

Specify the full duplex flow control to be done on the currently selected port. The options are XON/XOFF and NONE. The specified type of flow control will be done during both terminal emulation and file transfer. By default, XON/XOFF flow control is selected. XON/XOFF should not be used on half-duplex (local echo) connections, or when the other system does not support it. If XON/XOFF is used, HANDSHAKE should be set to NONE.

SET HANDSHAKE

Syntax: SET HANDSHAKE {CODE *number*, BELL, CR, LF, NONE, XOFF, XON}

Specify any half-duplex line turnaround handshake character to be used during file transfer on the currently selected port. The CODE *number* form allows any ASCII character to be specified by its decimal ASCII code. Handshake is NONE by default; if set to other than NONE, then FLOW-CONTROL should be set to NONE. In operation the handshake character is sought at the end of each received packet, following the normal END-OF-LINE character, but is not sent for outgoing packets.

SET INCOMPLETE

Syntax: SET INCOMPLETE {DISCARD, KEEP}

Specifies what to do with files that arrive incompletely: discard them or keep them. They are normally discarded.

SET INPUT

Syntax: SET INPUT {CASE, DEFAULT-TIMEOUT, ECHO, TIMEOUT-ACTION}

This command is described in Section 1.8, SCRIPTS.

SET KEY

Syntax: SET KEY *key-specifier* [*key-definition*]
Also: SET KEY {ON, OFF, CLEAR}

> *WARNING:* The format and functions of this command have changed substantially since version 2.29B and earlier. The changes were made in order to allow key redefinition to work on a wider variety of systems and keyboards without customization of the program source code for each configuration. See section 1.12 for further details.

Typical uses of SET KEY:

- You're used to having the ESC key in the upper left corner of the keyboard, but your new PC keyboard has an accent grave (``'') there. You can use SET KEY to make the accent key transmit an ESC, and you can assign accent grave to some other key.
- You send a lot of electronic mail, and always sign it the same way. You can put your "signature" on a single key to save yourself a lot of repetitive typing.
- You must set up your PC's function keys or numeric keypad to work properly with a host application.
- You have trouble with Kermit's 2-character escape sequences (like Ctrl-] C), and you want to assign these functions to single keys, like F10.

The SET KEY command does these things and more, and SHOW KEY gives us assistance. A key can be defined to:

- send a single character other than what it would normally send,
- send a string of multiple characters,
- invoke a CONNECT-mode Kermit action verb,
- send itself again.

SET KEY specifies that when the designated key is struck during terminal emulation, the specified character or string is sent or the specified Kermit action verb is performed. Key definitions operate only during CONNECT, not at Kermit-MS> or DOS command level.

The key-specifier is the identification of the key expressed in system-dependent terms. This can be a letter, such as Q for the key which produces an uppercase Q, or the numeric ASCII value of the letter in backslash notation (e.g.

‘‘\81’’), or else the numerical "scan code" observed by the system when the key is pressed (e.g. "\3856" for Ctrl-Alt-Shift-Q on an IBM PC). Material printed on keycaps is not necessarily a guide to what the key-specifier should be. When the word CLEAR is used in place of a key-specifier, all key definitions are cleared and then any built-in definitions are restored.

A string definition is one or more characters, including 8-bit values expressed in backslash form, such as

`SET KEY \315 directory\13`	IBM F1 key sends ‘‘directory<cr>’’
`SET KEY S X`	S key sends upper case X (a mean trick)
`SET KEY T \27[m`	T key sends three bytes: ESC [m
`SET KEY \2336 {del }xxx`	Alt-D sends "del "
`SET KEY \324 \Kexit`	F10 escapes back to Kermit-MS> prompt.

The string begins with the first non-spacing character following the key identification and continues until the end of line, exclusive of any trailing spaces. If a semicolon comment is used and the definition is given in a TAKE file, the line ends at the last non-spacing character before the semicolon. Curly braces, {...}, can be use to delimit the string in case you want the definition to include trailing spaces. All text after the closing bracket is ignored.

This manual does not contain a list of all the scan codes for all the keys on all the keyboards on all the PCs supported by MS-Kermit -- that would be a manual in itself. Rather, in order to obtain the key-specifier for the SET KEY command, you must type a SHOW KEY command and then press the desired key or key combination. This will report a scan code that you can use as the key specifier in a SET KEY command. To do this for many keys is a laborious process, so you should collect all your SET KEY commands into a file, which you can TAKE, or put them in your MSKERMIT.INI file.

If you enter SET KEY by itself, with no key specifier, the command will prompt you to press the selected key and again for the definition string. Certain characters, like ESC and CR, may not be entered literally into the string, but can be included by inserting escape codes of the form \nnn, a backslash followed by a 1- to 4-digit number corresponding to the ASCII value of the desired character. Where an ASCII digit follows directly after a backslash number, confusion can be avoided by placing curly braces {} around the backslashed number; thus, \{27}5 represents the two ASCII characters ESC and 5.

Here is an example of the use of SET KEY to assign ESC (ASCII 27) to the accent grave key. First the user gets the key-specifier for the key:

```
Kermit-MS>show key
  Push key to be shown (? shows all): ‘
  ASCII char: ‘ \96 decimal is defined as
  Self, no translation.
Free space: 129 key and 100 string definitions, 837 string characters.
```

The free space report says that 129 more keys may be redefined, and up to 100 of them may have multi-character strings assigned to them (as opposed to single characters), and that there are 837 bytes left for these strings, in total. Confident that there is enough space left for a new key definition, the user proceeds:

```
Kermit-MS>set key
  Push key to be defined: ‘
  Enter new definition: \27
```

Once a key definition is constructed and tested, it may be entered on a single line in a command file (such as MSKERMIT.INI):

```
set key \96 \27
```

To prevent accidents, SET KEY shows the current definition before asking for a new one; enter a Control-C to keep the current definition, or a carriage return to undefine the key, or a query mark (?) to see available choices.

The keyboard can be restored to its startup state, that is all redefinitions removed and all built-in defitions restored, by using the keyword CLEAR in place of the key identification:

```
SET KEY CLEAR
```

Undefined keys which do not send ASCII characters are trapped by the keyboard translator and are rejected; a beep results from using an undefined non-ASCII key.

SET KEY OFF directs MS-Kermit to read keycodes from DOS, rather than BIOS, so that console drivers like `ANSI.SYS` that operate at the DOS level may be used during Kermit CONNECT sessions. This would also apply to any special keyboard replacements that come with DOS-level drivers. SET KEY ON turns key definition back on, and returns Kermit to processing keystrokes at the BIOS level.

Kermit Action Verbs

An action verb is the shorthand expression for a named Kermit procedure, such as "generate the proper sequence for a left arrow," "show status," "send a BREAK," and others; verbs are complex actions and each verb has a name. In a key definition the verb name is preceeded by backslash K (`\K`) to avoid being confused with a string. Verbs and strings cannot be used together on a key.

```
SET KEY \331 \Klfarr
SET KEY \2349 \Kexit
```

makes the IBM keyboard left arrow key execute the verb named `lfarr` which sends the proper escape sequence for a VT102 left arrow key (which changes depending on the internal state of the VT102). The leading `\K` identifies the definition as a Kermit verb, so no string can start as `\K` or as `\{K` in upper or lower case (use `\92K`). The second example has Alt-X invoking the Leave-Connect-Mode verb "exit" (same as Kermit escape character ‘‘`^]`’’ followed by C).

Each system has its own list of verbs and predefined keys. Table 1-6 shows those available for the IBM PC family (there are also some additional verbs for reassigning Heath or VT100 function keys, see section 1.17.2). The SET KEY command shows the list of available verbs when a query mark `(?)` is given as a definition. SHOW KEY displays all currently defined keys or individually selected ones; SHOW KEY can be executed only interactively.

Some systems have preset key definitions when Kermit first begins (those for the IBM PC are shown in section 1.17.2). You can find out what they are on your system by typing SHOW KEY, and then question mark on the next line. You may supplement or change the predefined keys with SET KEY commands typed interactively or in `MSKERMIT.INI` or other command files.

The MS-Kermit CONNECT command may be used in conjunction with certain console drivers that do their own key redefinitions. Since MS-Kermit intercepts keystrokes at the BIOS level, drivers like `ANSI.SYS` which work at the DOS level will have no effect during CONNECT, even though they work at MS-Kermit command level. Other drivers, like SuperKey and ProKey, work at the BIOS level, and their key assignments will remain effective during Kermit terminal sessions, and additional Kermit SET KEY assignments may be made "on top" of them.

SET LOCAL-ECHO

Syntax: `SET LOCAL-ECHO {ON, OFF}`

Specify how characters are echoed during terminal emulation on the currently selected port. ON specifies that characters are to be echoed by Kermit-MS (because neither the remote computer nor the communications circuitry has been requested to echo), and is appropriate for half-duplex connections. LOCAL-ECHO is OFF by default, for full-duplex, remote echo operation.

Verb	Meaning
\Kupscn	Roll up (back) to previous screen
\Kdnscn	Roll down (forward) to next screen
\Khomscn	Roll up to top of screen memory
\Kendscn	Roll down to end of screen memory (current position)
\Kupone	Roll screen up one line
\Kdnone	Roll screen down one line
\Kprtscn	Print the current screen
\Kdump	Append the current screen to dump file
\Kholdscrn	Toggle hold screen mode
\Klogoff	Turn off session logging
\Klogon	Turn on session logging
\Ktermtype	Toggle terminal type
\Kreset	Reset terminal emulator to initial state
\Kmodeline	Toggle modeline off/on
\Kbreak	Send a BREAK signal
\Klbreak	Send a "long BREAK" signal
\Khangup	Drop DTR so modem will hang up phone
\Knull	Send a null (ASCII 0)
\Kdos	"Push" to DOS
\Khelp	Display CONNECT help message
\Kstatus	Display STATUS message
\Kterminals	Invoke user-defined macro TERMINALS, if any
\Kterminalr	Invoke user-defined macro TERMINALR, if any
\Kexit	Escape back from CONNECT mode
\Kgold,\Kpf1	VT102 keypad function key PF1
\Kpf2..\Kpf4	VT102 keypad function keys
\Kkp0..\Kkp9	VT102 keypad numeric keys
\Kkpdot,\Kkpminus,\Kkpcoma,\Kkpenter	Other VT102 keypad keys
\Kuparr,\Kdnarr,\Klfarr,\Krtarr	VT102 cursor (arrow) keys

Table 1-6: Kermit-MS Verbs for the IBM PC Family

SET MODE-LINE

Syntax: `SET MODE-LINE {ON, OFF}`

On systems, like the IBM PC family, which are capable of displaying a status, or "mode" line on the 25th (or bottom) line during terminal connection, disable or enable this function. This command has no effect on systems that do not display a mode line during connect.

The mode line shows several important facts about the connection, like which port is being used, the transmission speed and parity, the current escape character, etc. When the mode line is enabled, it may be turned on and off using the CONNECT escape-level command M or the Kermit verb "modeline".

The mode line occupies the 25th line of those systems that have such a thing, and is not affected by scrolling (on some systems that have large screens, the mode line should appear on whatever the bottom line is, e.g. the 43rd). When emulating a VT102 or Heath-19, Kermit will allow the host to address the 25th line directly using cursor positioning commands. If this happens, Kermit will remove its mode line and relinquish control of the 25th line to the host (as if you had typed SET MODE OFF). When the Tektronix, or no terminal at all, is being emulated, the 25th line (if any) is available for scrolling. If the mode line is disabled by an application or by the command SET MODE OFF then the only way to revive Kermit's mode line display is to give the command SET MODE ON.

SET PARITY

Syntax: `SET PARITY {EVEN, ODD, MARK, SPACE, NONE}`

Specify the character parity to be used on the currently selected port. You will need to SET PARITY to ODD, EVEN, MARK, or possibly SPACE when communicating with a system, or over a network, or through modems, concentrators, multiplexers, or front ends that require or impose character parity on the communication line. For instance, most IBM mainframe computers use EVEN or MARK parity; Telenet normally uses MARK parity. If you neglect to SET PARITY when the communications equipment requires it, the symptom may be that terminal emulation works (well or maybe only partially), but file transfer or script INPUT commands do not work at all.

NONE means that no parity processing is done, and the 8th bit of each character can be used for data when transmitting binary files. This is the normal case. If parity is other than none, then there will be 7 data bits (use of parity with 8 data bits is not supported).

If you have set parity to ODD, EVEN, MARK, or SPACE, then Kermit-MS will request that binary files be transferred using 8th-bit-prefixing. If the other Kermit knows how to do 8th-bit-prefixing (this is an optional feature of the Kermit protocol, and some implementations of Kermit don't have it), then 8-bit binary files can be transmitted successfully. If NONE is specified, 8th-bit-prefixing will not be requested. Note that there is no advantage to using parity. It reduces Kermit's file transfer efficiency without providing additional error detection. The SET PARITY command is provided only to allow Kermit to adapt to conditions where parity is required, or 8-bit transmission is otherwise thwarted.

If parity is in use, then the display during terminal emulation, as well as any session log, will be 7-bit ASCII, unless you have SET DEBUG ON (q.v.).

There may be situations in which you require 7-bit ASCII with no parity during terminal emulation, but still want to force 8th bit prefixing during file transfer. To accomplish this, SET PARITY SPACE.

The INPUT and TRANSMIT commands use 7 or 8 bits if parity is NONE, according to the SET DISPLAY command, and this may upset recognition of received characters when the host unexpectedly sends them with its own parity.

WARNING: The SET PARITY command has no effect on a port used for printing. This is because printing is done by DOS, not Kermit. Since Kermit clears hardware parity on COM1 at startup, it is not recommended that COM1 be used for a serial printer, unless the printer works with no parity.

SET PORT

Syntax: `SET PORT {`*number*`, COM`*n*`, BIOS`*n*`, NET [`*nodename*`], UB-NET1 [`*nodename*`]}`

On machines with more than one communications port, select the port to use for file transfer and CONNECT. This command lets you use a different asynchronous adapter, or switch between two or more simultaneous remote sessions. Subsequent SET SPEED, PARITY, HANDSHAKE, FLOW, and LOCAL-ECHO commands will apply to this port only -- each port remembers its own parameters, so that you may set them for each port and then switch between ports conveniently with the SET PORT command.

SET PORT 1 selects COM1, SET PORT 2 selects COM2. All versions default to port 1, except for the IBM PCjr, which uses port 2 if its internal modem is installed. Additionally, COM3 and COM4 are supported for IBM PC/AT's and PS/2's, as explained in Section 1.18.3.

SET PORT BIOS*n*, on machines which support it, instructs Kermit to do serial port input and output by Bios calls rather than going directly to the hardware (*n* is a digit between 1 and 4). The most important use is allowing selected network packages to intercept such Bios calls and relay the characters across the network.

In "generic" MS-DOS Kermit, the following alternate forms allow you to experiment with device names or numbers until you find the communication port:

```
SET PORT {DEVICE, FILE-HANDLE}
```

Just type a carriage return after either of these commands, and you will be prompted for a device name or a numeric port-handle. Keep trying till you find one that works. File-handle 3, the system auxillary device, is conventional on many machines, as are device names COM1, COM2, and AUX.

MS-Kermit for the IBM PC family is able to operate over local area networks through the NetBIOS interface. The command

SET PORT NET [*nodename*]

redirects communications the LAN board installed in the local computer and the associated NetBIOS emulator software, if active, rather than the serial port or the COM device driver. It installs a unique Kermit node name in the local LAN, so that other nodes can refer to it when files are transferred or terminal emulation is done. This name is displayed when you give the SET PORT NET command. The server should use SET PORT NET, and the client should use SET PORT NAME *nodename*, specifying the server's name, e.g. `mskermit.K`. Note that alphabetic case is significant in node names!

Both the regular serial port and a network connection can be kept alive simultaneously; clearly, only one can be used at a time under MS-DOS. MS-DOS 3.x is not required for Kermit network usage, but most LANS do need DOS 3.1 or later for conventional file server work. Kermit needs only the NetBIOS emulator network software.

SET PORT UB-NET1 is implemented on the IBM PC version of Kermit to allow connection to Ungermann-Bass Net One LAN NETCI interface and behaves similarly to the NetBIOS method.

SET PROMPT

Syntax: SET PROMPT [*string*]

This command allows you to change the MS-DOS Kermit program's prompt. The string may be enclosed in curly braces. Control characters like ESC can be included as backslashed numbers like ‘‘`\27`’’. `ANSI.SYS` and similar console drivers can be programmed through this command to get a boldface, inverse, and/or blinking prompt. The prompt string must be less than 128 characters. If the string is omitted (missing) Kermit's original prompt of ‘‘`Kermit-MS>`’’ is restored.

SET RECEIVE

Syntax: SET RECEIVE *parameter value*

This command lets you modify the ways in which MS-Kermit asks the other Kermit to behave. That is, it controls the file transfer protocol options for packets sent to MS-Kermit by the other Kermit. The parameters and values you specify in the SET RECEIVE command are sent to the other Kermit during initial negotiations. Numbers may be specified as ordinary decimal numbers (74), or in backslash notation (\x03F).

END-OF-LINE *number*
: The ASCII value of terminating character to look for on incoming packets. Normally carriage return. Use this command if the other Kermit is terminating its packets with some other control character.

PACKET-LENGTH *number*
: Ask the remote Kermit to use the specified maximum length for packets that it sends to Kermit-MS. The normal length is 94 bytes. Use this command to shorten packets if the communication line is noisy or terminal buffers somewhere along the path are too small. Shorter packets decrease the probability that a particular packet will be corrupted, and will reduce the retransmission overhead when corruption occurs, but will increase the file transfer throughput.

If a length greater than 94 is specified, a protocol option called "long packets" will be used, provided the other Kermit also supports it. Kermit-MS can receive extended-length packets up to 1000 bytes long. Long Packets can improve efficiency by reducing the per-packet overhead for a file, but they will not be used unless you issue this command. Before using this option, ensure that the equipment on the communications pathway can absorb a long packet, and that the connection is clean (retransmission of long packets is expensive!). You should also SET BLOCK-CHECK 2 or 3 for more reliable error checking.

`PADCHAR` *number*
: Ask the remote Kermit to use the given control character (expressed as a decimal number 0-31, or 127) for interpacket padding. Kermit-MS should never require any padding.

`PADDING` *number*
: Ask the remote Kermit to insert the given number of padding characters before each packet it sends. MS-Kermit never needs padding, but this mechanism might be required to keep some intervening communication equipment happy.

`START-OF-PACKET` *number*
: If the remote Kermit will be marking the beginning of packets with a control character other than Control-A, use this command to tell Kermit-MS about it (the number should be the decimal ASCII value of a control character). This will be necessary only if the hosts or communication equipment involved cannot pass a Control-A through as data, or if some piece of communication equipment is echoing packets back at you.

`TIMEOUT` *number*
: Ask the remote Kermit to time out and retransmit after the given number of seconds if a packet expected from Kermit-MS has not arrived. Use this command to change the other Kermit's normal timeout interval.

SET REMOTE

Syntax: `SET REMOTE {ON, OFF}`

SET REMOTE ON removes the file transfer display (as if you had given the command SET DISPLAY QUIET). It should be used when you are running Kermit-MS in remote mode when coming in from another PC through the Kermit-MS's "back port", to which the console has been reassigned using the DOS CTTY command, e.g.

```
CTTY COM1
```

It is necessary to issue the SET REMOTE ON command because (a) Kermit-MS has no way of knowing that its console has been redirected, and (b) when the console is the same as the port, the file transfer display will interfere with the file transfer itself. SET REMOTE OFF returns the file transfer display to its preferred style (REGULAR or SERIAL). When you SET REMOTE ON, you might also want to SET DELAY 5 or thereabouts, to allow yourself time to escape back to the local system before MS-Kermit starts sending packets.

On the IBM PC, CTTY CON returns control to the normal keyboard and screen (other systems may use other device names, e.g. SCRN). See section 1.18.4 for further details about remote operation.

If you are using a port other than COM1 on the remote MS-Kermit, you must give it an appropriate SET PORT command.

WARNING: During CTTY console redirection, many programs still output to the real screen and require input from the real keyboard and will hang the system until keyboard requests are satisfied.

SET RETRY

Syntax: `SET RETRY` *number*

Sets the number of times a packet is retransmitted before the protocol gives up. The number of retries can be between 1 and 63, and is 5 by default. This is an especially useful parameter when the communications line is noisy or the remote host is very busy. The initial packet of a file exchange is given three times as many retries to allow both systems to become ready.

SET SEND

Syntax: `SET SEND` *parameter value*

The SET SEND command is used primarily to override negotiated protocol options, or to establish them before they are negotiated.

`END-OF-LINE` *number*
: ASCII value of packet terminator to put on outbound packets. Normally carriage return. Use this command if the other Kermit needs its packets terminated with a nonstandard control character.

`PACKET-LENGTH` *number*
: Use this as the maximum length for outbound packets, regardless of what the other Kermit asks for. Normally, you would use this command only to send shorter packets than the other Kermit requests, because you know something the other Kermit doesn't know, e.g. there's a device on the communication path with small buffers.

`PADCHAR` *number*
: Use the specified control character for interpacket padding. Some hosts may require some padding characters (normally NUL or DEL) before a packet, and certain front ends or other communication equipment may need certain control characters to put them in the right modes. The number is the ASCII decimal value of the padding character, (0 - 31, or 127).

`PADDING` *number*
: How many copies of the pad character to send before each packet, normally zero.

`PAUSE` *number*
: How many milliseconds to pause before sending each packet, 0-127, normally zero. This may help half-duplex or slow systems prepare for reception of our packet. Padding characters are sent only after the time limit expires.

`QUOTE` *number*
: Use the indicated printable character for prefixing (quoting) control characters and other prefix characters. The only reason to change this would be for sending a very long file that contains very many ‘‘#’’ characters (the normal control prefix) as data.

`START-OF-PACKET` *number*
: Mark the beginning of outbound packets with some control character other than Control-A. This will be necessary if the remote host or the communication channel cannot accept a Control-A as data, or if it echoes back your packets. The remote host must have been given the corresponding SET RECEIVE START-OF-PACKET command.

`TIMEOUT` *number*
: Change Kermit-MS's normal timeout interval; this command is effective only if TIMER is set to be ON; it is normally ON, with a default interval of 13 seconds.

SET SERVER

Syntax: `SET SERVER TIMEOUT` *seconds*

Specify how often the MS-DOS Kermit server should send NAK packets while waiting for commands. These NAK packets are used to recover from deadlocks that might occur when the other Kermit sends an initial packet which is lost, but does not have the capability to time out and retransmit it. These NAKs can be supressed entirely by specifying a value of zero. This may be necessary to avoid interfering with certain modems or PBXs that go into originate mode when they receive input from the PC, when in fact you want the device to be in answer mode.

SET SPEED

Syntax: `SET SPEED` *rate*

Set the transmission speed (in bits per second, commonly called *baud*) of the currently selected terminal communications port to 300, 1200, 1800, 2400, 4800, 9600, or other common speed, and on the IBM PC family, higher speeds including 19200, 38400, 57600, and 115200. Both connected systems, as well as any intervening communication equipment, must be able to support the specified transmission speed, and both systems should be set to the same speed.

Some implementations do not support the SET SPEED command. But Kermit-MS leaves the current communication port settings alone unless you issue explicit SET commands to change them, so you may use MODE or other DOS programs to establish the desired settings before running Kermit.

On certain systems, when you first run Kermit after powering the system up, you may get a message "Unrecognized baud rate". This means that Kermit tried to read the baud rate from the port and none was set. Simply use SET SPEED (if available) or the DOS MODE command to set the desired baud rate.

SET BAUD is a synonym for SET SPEED.

SET TAKE-ECHO

Syntax: `SET TAKE-ECHO {ON, OFF}`

Specifies whether screen display should occur during implicit or explicit TAKE operations on `MSKERMIT.INI` or other Kermit-MS command files, and during evaluation of macro definitions by the DO command. Handy for finding errors in TAKE files or macro definitions.

SET TERMINAL

Syntax: `SET TERMINAL {`*type*`,` *parameter* `[`*value*`]}`

This command controls most aspects of terminal emulation. Most of the parameters are only settable (or meaningful) on the IBM PC family and compatibles. (Programmers who are proficient on other MS-DOS systems are invited to fill in these functions for those systems and send the results back to Columbia.) On other systems, built-in setup modes or DOS commands can be used to accomplish the same functions.

The first group of parameters tells which kind of terminal to emulate. When Kermit-MS uses its built-in software for emulation, incoming characters are examined for screen control commands (escape sequences) specific to that terminal, and if encountered, the commands are executed on the PC screen.

`NONE` Act as a dumb terminal. All incoming characters will be sent to the screen "bare", as-is, through DOS. If you have loaded a device driver into DOS for the `CON` device, such as `ANSI.SYS`, then that driver will be able to interpret the codes itself. Many non-IBM systems have their own screen control code interpreter built into DOS or firmware, or available as a loadable device driver.

`VT52` The DEC VT-52 terminal.

`HEATH` The Heath/Zenith-19 terminal (H19), which supports all the VT52 commands, plus line and character insert/delete editing functions, an ANSI mode, and a 25th line.

`VT102` The DEC VT102 (ANSI) terminal, which is the same as the VT100 but also supports line/character insert/delete editing functions and ANSI printer controls.

`TEK4010`
A Tektronix 4010 graphics terminal. Currently only available on IBM, TI, and Victor PCs. On the IBM family, Kermit automatically senses and adapts to the CGA, EGA, Monochrome, Hercules, or ATT style board.

On the IBM family, you may "toggle" among the supported terminal emulations by typing Alt-Minus.

The specific escape sequences supported by Kermit for each of these terminal types are listed in section 1.17.1. Note that when a Kermit program includes Tektronix emulation, this can be invoked automatically while in character mode (VT102, VT52, or Heath emulation) when the emulator receives certain escape sequences. This can be turned off using the DISABLE TEK command.

The remaining SET TERMINAL commands specify setup options for the selected terminal:

`CHARACTER-SET {UK, US, ALTERNATE-ROM}`
: UK displays ‘‘#’’ (ASCII 35, number sign) as a pound sterling sign, US displays ‘‘#’’ as ‘‘#’’. ALTERNATE-ROM maps accent grave and the lowercase letters to be national characters in the IBM video adapter. That is, character codes of 60h to 7Ah (accent grave, lower case a-z) are mapped to codes 80h to 9Ah. The SET TERMINAL CHARACTER-SET command applies only during VT100/102 emulation.

`CLEAR-SCREEN`
: Clears the screen, so that a subsequent CONNECT command shows a blank screen. The action taken is identical to Kermit’s `\Kreset` verb.

`COLOR` *number* `[,` *number* `[,` *number*`]]`
: Several numbers, applied in left to right sequence, separated by commas or spaces:

0	Reset the colors to normal intensity white characters on a black background and use the "no-snow" mode on the IBM Color Graphics Adapter (CGA).
1	High intensity foreground
10	Request fast screen updating for use on the IBM Mono, EGA, or VGA (usually sensed and set internally by Kermit), and some non-IBM CGAs.
3*x*	Foreground color
4*x*	Background color

where *x* is a single digit from 0 to 7, which is the sum of the desired colors:

1	Red
2	Green
4	Blue

Example: "SET TERMINAL COLOR 0 1 37 44" on an IBM CGA would produce bold white characters on a blue field with no snow. The snow removal business has to do with whether the program should synchronize with vertical retrace when updating screen memory. This is necessary with certain color adaptors (like the CGA) and unnecessary for others (like the EGA).

`CURSOR-STYLE {BLOCK, UNDERLINE}`
: Sets the cursor rendition to your preference. Note that on some early IBM PCs and compatibles, the cursor may not be restored correctly after escaping back from CONNECT because of a bug in the early IBM BIOS.

`DIRECTION {LEFT-TO-RIGHT, RIGHT-TO-LEFT}`
: Controls the direction of screen display during CONNECT. You may use Right-to-Left for Hebrew or Arabic, provided you have the appropriate character sets loaded.

`KEYCLICK {ON, OFF}`
: Turns electronic keyclick ON or OFF. If your keyboard has a mechanical clicker (as IBM boards do), you may not notice the effect of this command.

`GRAPHICS {AUTO-SENSING, CGA, EGA, VGA, HERCULES, ATT}`
: Manually selects the kind of display adapter for Tektronix graphics. AUTO-SENSING is the default, VGA means 640x480x16 colors, and ATT encompasses the ATT 6300 series, Olivetti M24/M28, DEC VAXmate II, and the Toshiba T3100 in 640x400 b/w (see Table 1-5).

`MARGIN-BELL {ON, OFF}`
: Controls whether the bell should be sounded when the cursor passes column 72 near the right screen margin; wider displays set the bell 8 columns from the right edge.

`NEWLINE-MODE {ON, OFF}`
ON sends a carriage-return-linefeed combination (CRLF) when you type carriage return (CR) during terminal emulation. OFF (default) just sends a CR when you type CR. Useful in conjunction with SET LOCAL-ECHO ON when CONNECTing two PC's back-to-back.

`ROLL {ON, OFF}`
ON unrolls the screen to the bottom before adding new material if the screen had been rolled back, e.g. by Ctrl-PgUp. ROLL OFF (the default) displays new material on the current screen, possibly overwriting old material.

`SCREEN-BACKGROUND {NORMAL, REVERSE}`
NORMAL means dark background, light characters. REVERSE means light background, dark characters.

`TAB {AT` *n*`, CLEAR AT` *n*`, CLEAR ALL}`
Sets tab stops or clears one or all tab stops; *n* is the numeric position of the tab to be set or cleared. By default, tabs are every 8 spaces, at positions 9, 17, 25, etc. Only meaningful when emulating a terminal that has settable tabs (the VT52 doesn't really but the emulator can set them anyway). More than one tabstop may be specified by separating column numbers with commas, spaces, or tabs. You may also use the notation "*m*:*n*" to specify regularly spaced tabs across the screen, where *m* is the initial tab position, and *n* is the spacing between tabs. 132 columns are supported.

`WRAP {ON, OFF}`
ON automatically breaks screen lines (by inserting a CRLF) when they reach the right margin. OFF disables wrapping -- if a line is too long, the excess characters go off the screen. WRAP is OFF by default, since most hosts format lines to fit on your screen.

SET TIMER

Syntax: `SET TIMER {ON, OFF}`

This command enables or disables the timer that is used during file transfer to break deadlocks that occur when expected packets do not arrive. By default, the timer is ON. If the other Kermit is providing timeouts, you can safely turn the timer OFF to avoid unnecessary retransmissions that occur when two timers go off simultaneously.

SET TRANSLATION

Syntax: `SET TRANSLATION INPUT {ON, OFF,` *char1 char2*`}`

This command provides multi-language support (and perhaps other special effects) during CONNECT, and during execution of the INPUT, OUTPUT, PAUSE, and TRANSMIT script commands, but not during file transfer or at MS-Kermit command level. A character that arrives at the communication port (*char1*) will be translated to another character (*char2*) before display on the screen. As many as 256 characters may have translations specified concurrently. But to see characters with ASCII values higher than 127, you must also SET DISPLAY 8 and SET PARITY NONE.

SET TRANSLATION INPUT ON enables translation (the keyword INPUT is required to allow future translation mechanisms). OFF disables the translation and is the default. So even if you have set up a translation table, you must SET TRANSLATION INPUT ON before it will take effect. SHOW TRANSLATION tells whether translation is OFF or ON, and displays any current table entries.

Translation table entries are made by specifying byte pairs in ASCII or numeric backslash form:

```
SET TRANS INPUT \3 \13
```

converts incoming ASCII ETX characters (decimal 3) to ASCII CR (decimal 13). 8-bit values are allowed, and refer to characters in the "upper half" of the PC's character set, either the ROM characters supplied with the PC or else substitutions provided by a special device driver.

A more practical example shows how the user of a German PC could use the SET TRANSLATION and SET KEY commands to make the PC's umlaut-a key (key code 132) send a left curly brace (‘‘{’’, ASCII 123), and to display incoming curly braces as umlaut-a's:

```
SET KEY \d132 \d123
SET TRANS INP { \d132
```

(This example applies to the IBM PC German keyboard, and assumes the German keyboard driver, KEYBGR, has been loaded. This is usually done in `AUTOEXEC.BAT`.)

SET WARNING

Syntax: `SET WARNING {ON, OFF}`

Specify what to do when an incoming file is about to be stored under the same name as an existing file in the target device and directory. If ON, Kermit will warn you when an incoming file has the same name as an existing file, and automatically rename the incoming file (as indicated in the warning message) so as not to destroy (overwrite) any existing one. If OFF, the pre-existing file is destroyed, even if the incoming file does not arrive completely. WARNING is ON by default as a safety measure, and the current setting may be observed in the SHOW FILE display.

The new name is formed by adding numbers to the part of the name before the dot. For instance, `ABC.TXT` becomes `ABC00001.TXT`, `ABC00001.TXT` becomes `ABC00002.TXT`, etc. If the name already has eight characters, then digits replace the rightmost characters.

1.6.11. The STATUS and SHOW Commands

The values of MS-Kermit options that can be SET, DEFINEd, ENABLEd, or DISABLEd can be displayed using the STATUS or SHOW commands.

The STATUS Command

Syntax: `STATUS`

The STATUS command displays the values of the current SET options on a single screen. There are no operands for the STATUS command. Use the SHOW command to see logically-grouped settings, e.g. SHOW COMMUNICATIONS, SHOW TERMINAL.

The SHOW Command

Syntax: `SHOW` *option*

The SHOW command is used for displaying communication parameters, protocol settings, macro definitions, key redefinitions, file transfer statistics, translations, and other common groupings.

`SHOW COMMUNICATIONS`
: displays the settings of the current serial port (port, speed, parity, echo, etc) and the status of modem signals Carrier Detect, Data Set (modem) Ready, and Clear To Send.

`SHOW FILE`
: displays the file transfer control settings, such as the current path, file discard, attributes packets on/off, warning, end-of-file convention, etc.

`SHOW KEY`
: allows you to determine a key's identification code and what it will send in CONNECT mode, most useful for obtaining the identification of a key when SET KEY commands will be placed in a TAKE file. This command can be done only interactively (use a `?` to see all defined keys). Refer to the SET KEY description for details.

`SHOW LOGGING`
: Displays the names of the session, packet, and transaction logs, and tells whether logging is in effect.

`SHOW MACROS [macroname]`
displays the definitions of all currently defined macros, as well as the amount of space left for new macro definitions. A macro name, or abbreviation, can be included to restrict the list, e.g. SHOW MACRO IBM will display the definition of the IBM macro, and SHOW MACRO X will list the definitions of all macros whose names begin with X.

`SHOW MODEM`
displays the status of the modem signals DSR (dataset ready, modem tells the PC that it is turned on and in data mode), CTS (clear to send, modem grants the PC permission to send data), and CD (carrier detect, local modem tells the PC that it is connected to the remote modem). The results may be misleading if your asynchronous adapter, or the connector or cable that is attached to it, is strapped to supply these modem signals itself.

`SHOW PROTOCOL`
displays the values of the Kermit protocol-related parameters, including all the SET SEND and SET RECEIVE parameters, plus whether the timer, attribute packets, and logging are enabled.

`SHOW SCRIPTS`
displays the script-related variables.

`SHOW SERVER`
displays which server functions are enabled and disabled.

`SHOW STATISTICS`
displays counts of characters sent and received during file transfers, for both the most recent transfer and the entire session, and an estimate of the average baud rate while sending and listening.

`SHOW TERMINAL`
displays the terminal settings, which terminal is being emulated, the tab stops, etc.

`SHOW TRANSLATION`
displays the entries in the 256 byte input translation table. Values are expressed numerically to avoid confusion with different display adapters, and the command shows only entries for which input and output codes differ.

1.7. Macros

Like TAKE files, macros provide a way of collecting many commands into a single command. The difference between a macro and a TAKE file is that Kermit keeps all its macro definitions in memory, and can execute them as many times as you like, without having to look them up on disk, whereas every time you issue a TAKE command, Kermit has to access a disk. But . . . you can have as many TAKE command files as you like, and they can be as long as you want, whereas MS-Kermit's memory for storing macro definitions is limited. You can put macro definitions and DO commands for them in TAKE files, or for that matter, you can put TAKE commands in macro definitions. There is a limit of 25 simultaneously active TAKE files plus active macros; a TAKE file or macro remains active if the last item invokes another TAKE or macro command. Active here means Kermit is reading commands from them, not just storing them for later.

The DEFINE Command

Syntax: `DEFINE` *macro-name* `[`*command* `[,` *command* `[, ...]]]`

Kermit-MS command macros are constructed with the DEFINE command. Any Kermit-MS commands may be included. Example:

```
define telenet set parity mark, set speed 1200, connect
```

A macro can be undefined by typing an empty DEFINE command for it, like

```
define telenet
```

A macro definition may be up to 255 character long. This example shows a long definition in which lines are continued with hyphenation:

```
define setup set port 1, set speed 19200, set parity even,-
set flow none, set handshake xon, set local-echo on,-
set timer on, set terminal color 1 31 45,-
set warning on, set incomplete keep, connect
```

Longer definitions can be accomplished by "chaining." Example:

```
define setup set port 1, set speed 19200, set par even, do setup2
define setup2 set flo no, set handsh xon, set local on, do setup3
define setup3 set timer on, set terminal color 1 31 45, do setup4
define setup4 set warning on, set incomplete keep, connect
```

DO SETUP or just SETUP will invoke all of these commands. Commas are used to separate commands in macro definitions; carriage returns (`\13`) cannot be used. When control or other special characters are needed in a macro they may be expressed in backslash number form, `\`*nnn*.

The SHOW MACROS command displays the values of currently defined macros, and tells how much space is left for further definitions.

The definition of the macro is entered literally; variables are not evaluated (see ASSIGN, below).

The DO Command

Syntax: `[DO]` *macro-name* `[`*parameters...*`]`

A Kermit-MS macro is invoked using the DO command. For instance, Kermit-MS comes with a predefined macro to allow convenient setup for IBM mainframe line-mode communications; to invoke it, you would type DO IBM. The IBM macro is defined as "set timer on, set local-echo on, set parity mark, handshake xon, set flow none". You can use the DEFINE command to redefine this macro or remove the definition altogether.

There is no automatic way to undo the effect of a macro. If you need to accomplish this effect, you should define another macro for that purpose. For instance, to undo the effect of "do ibm" so that you could connect to, say, a DEC VAX, you could:

```
def vax set parity none, set handshake none, set flow xon/xoff,-
set timer off, set local-echo off
```

Then you can "do ibm" whenever you want to use the IBM system, and "do vax" whenever you want to use the VAX.

If you wish to view the macro expansion whenever you issue a DO command, you can SET TAKE-ECHO ON.

As a convenience the word DO may be omitted. However, when question-mark help is sought at the Kermit prompt, only the main keyword help table will be shown. If you want to see the available macros, type "do ?" or SHOW MACROS. Use of DO is recommended for overall clarity unless a favorite macro is executed frequently.

Variables

Macros can use substitution variables similar to those of DOS Batch. The name of a substitution variable is of the form "`\%`*character*", where the single character is a digit or a letter or other 8-bit character whose ASCII value is 48 decimal or larger; upper and lower case letters are considered to be the same character. A substitution variable is defined as a string of text by the DEFINE command (the variables are in fact macros) and Kermit replaces occurrences of the variable name with that text, hence the word "substitution". For example,

```
Kermit-MS>define \%a this is substituted material
Kermit-MS>echo I wonder if \%a or not.
```

yields the display:

```
I wonder if this is substituted material or not.
```

Another example:

```
Kermit-MS>define \%c set port 1,set speed 9600,set parity even,connect
```

Then

```
Kermit-MS>\%c
```

is equivalent to

```
Kermit-MS>set port com1
Kermit-MS>set speed 9600
Kermit-MS>set parity even
Kermit-MS>connect
```

The special subset of substitution variables, `\%1 .. \%9`, is similar to the DOS Batch variable set `%1 .. %9`. The DO command can accept arguments after the macro name and the individual words in the arguments become the definitions of `\%1`, etc, for up to nine words, in order. For example, given the following definition:

```
def dial ATDT\%1\13,input 30 CONNECT,connect,in Login:,out \%2\13
```

the following command can be used to dial any phone number:

```
Kermit-MS>do dial 555-1212 myname
```

The word DO may be omitted, as in:

```
Kermit-MS>dial 555-1212 myname
```

This command automatically assigns the value "555-1212" to variable the `\%1` and "myname" to `\%2`, and uses these values while dialing the phone and logging into the host system. If fewer than nine words are seen the remaining variables are not changed. For example, if the line above was busy, you could dial a different number and omit the username because it will be remembered from last time.

If it is desired to assign multiple words to a single variable, they can be grouped in braces, for example

```
Kermit-MS>dial {212 555 1212} myname
```

Substitution variables can reference other substitution variables in their definitions. Care is needed to prevent circular definitions, but even those are detected by Kermit. Subtle circular executions could cause Kermit to go into an endless loop; if you think this is happening, type a Control-C to interrupt the process. To clarify matters, the definition string of a variable is substituted for the variable's name when the name is observed in a left to right scan of a command. For example,

```
Kermit-MS>define \%a echo This is \%b example: \%b.
Kermit-MS>define \%b a mac\%c expansion
Kermit-MS>define \%c ro string
Kermit-MS>\%a
```

displays:

```
This is a macro string expansion example: a macro string expansion.
```

If this example is entered manually then when the final `\%a` is typed the command line is immediately replaced with the fully expanded command and more input is solicited (such as a carriage return). Try it. Check the variable definitions with the SHOW MACRO command.

A variable can be undefined (deleted) by defining it as an empty string:

```
Kermit-MS>define \%c
```

DOS batch file arguments may be transformed into Kermit variables. Suppose file `TEST.BAT` holds the line:

```
Kermit define \%%1 %1, define \%%a %2, stay
```

Invoking the Batch file by:

```
C>test one two
```

results in creating Kermit variables `\%1` with definition of "one" and `\%a` with definition "two". The doubled percent symbols in the Batch file are needed to compensate for one of them being consumed by the DOS Batch processor. `%1` is the first Batch argument word, `%2` is the second word. The syntax `\%%1` is converted by Batch to be `\%1` when seen by Kermit, without further substitution by Batch.

The ASSIGN Command

Syntax: `ASSIGN`

The DEFINE command does not evaluate the definition. For instance, the command

```
define \%a \%1
```

simply defines the variable `\%a` to be ''`\%1`'', not the current *value* of `\%1` -- if `\%1` changes, then so does `\%a`. To copy the *value* of one variable to another, use the ASSIGN command:

```
assign \%a \%1
```

This copies the value of `\%1` to `\%a`, so that if `\%1` changes, `\%a` will retain the previous value. Example:

```
Kermit-MS>define \%a foo
Kermit-MS>define \%b \%a
Kermit-MS>echo \%a \%b
foo foo
Kermit-MS>assign \%c \%a
Kermit-MS>define \%a new
Kermit-MS>echo \%a \%b
new new
Kermit-MS>echo \%a \%c
new foo
```

1.8. SCRIPTS

A script is a file or a macro containing Kermit commands to be executed. What distinguishes a script from ordinary TAKE files or macros is the presence of INPUT, REINPUT, OUTPUT, PAUSE, ECHO, ASK, CLEAR, IF, GOTO, and WAIT commands to automatically detect and respond to information flowing though the serial port, actions which otherwise would be performed by the user during CONNECT. The login sequence of a host computer is a classical example.

It is a common, but incorrect, assumption that text to be sent to the remote computer can be included in a TAKE file after the CONNECT command:

```
set speed 9600        ; MS-Kermit command
connect               ; MS-Kermit command
run kermit            ; Text to be sent to other system
send foo.bar          ; Text to be sent to other system
^]c                   ; Escape sequence to get back to MS-Kermit
receive               ; MS-Kermit command
```

The reason this doesn't work is that during CONNECT, MS-Kermit always reads from the real keyboard, and not from the take file. Even if this technique did work, it would still run into synchronization problems. But these can be avoided when there is a way to coordinate the commands that we send with the remote system's responses. Kermit's script commands provide this ability. They may be freely intermixed in a TAKE file or macro with any other Kermit commands to achieve any desired effect. The OUTPUT command sends the specified characters as if the user had typed them; the INPUT command reads the responses and compares them with specified character strings, just as the user would do.

The script commands include INPUT, REINPUT, OUTPUT, PAUSE, WAIT, ECHO, IF, ASK, and GOTO. These commands may be interrupted by typing Ctrl-C at the keyboard. The INPUT, REINPUT, PAUSE, and WAIT commands accept a following number as a timeout value. The number is interpreted as seconds from the present or, if given in `hh:mm:ss` form, as a specific time of day. In either case, the timeout interval must be within 12 hours of the present to avoid it being considered as in the past (expired).

HINT: It is recommended that a console driver such as `ANSI.SYS` be loaded during executing of a script. This is because Kermit's terminal emulator is active only during the CONNECT command, and any PC/host interactions that occur during script execution may appear fractured on the screen. This is particularly true of full-screen login applications, like through an IBM 3270 protocol converter.

The CLEAR Command

Syntax: `CLEAR`

The CLEAR command empties the buffers of the serial port to forget any earlier material. This gets the INPUT command off to a clean start. (This command was called CLRINP in 2.29B and earlier, and CLEAR was used to erase macro and key definition memory).

The ECHO Command

Syntax: `ECHO` *text*

The ECHO command is useful for reporting progress of a script, or prompting the user for interactive input. The text is displayed on the screen, and may include backslash notation for control or 8-bit characters. An implied linefeed is included at the beginning of the text.

SET INPUT

Syntax: `SET INPUT {CASE, DEFAULT-TIMEOUT, ECHO, TIMEOUT-ACTION}`

The SET INPUT command controls the behavior of the script INPUT command:

`SET INPUT CASE {IGNORE, OBSERVE}`
: Says whether or not to distinguish upper and lower case letters when doing a matchup in the INPUT command. OBSERVE causes upper and lower case letters to be distinguished. The default is to IGNORE case distinctions.

`SET INPUT DEFAULT-TIMEOUT` *seconds*
: Changes the default waiting time from one second to this new value. The value is used when an INPUT command has no timeout specified.

`SET INPUT ECHO {ON, OFF}`
: Show on the screen characters read from the serial port during the script operation, or not. Default is ON, show them.

`SET INPUT TIMEOUT-ACTION {PROCEED, QUIT}`
: Determines whether or not the current macro or TAKE command file is to be continued or exited if a timeout occurs. PROCEED is the default and means that timeouts are ignored. QUIT causes the current script file to be exited and control passed to either the next higher level script file (if there is one) or to Kermit's main prompt.

The SHOW SCRIPTS command displays the SET INPUT values.

The INPUT command

Syntax: `INPUT [`*timeout*`] {`*search-string*`, @`*filespec*`}`

INPUT is the most powerful of the script commands. It reads characters from the serial port continuously until one of two things occurs: the received characters match the search string or the time limit expires. Matching strings is the normal use, as in:

```
Kermit-MS>input 5 Login please:
```

to recognize the phrase "Login please:", or else time out after waiting for 5 seconds. A special binary character `\255` or `\o377` or `\xFF` stands for the combination carriage return and a line feed, in either order, to simplify pattern matching. The command reports a testable status of SUCCESS or FAILURE and sets the DOS ERRORLEVEL parameter to 2 if it fails to match within the timeout interval. Characters are stored in a 128 byte buffer for later examination by REINPUT, discussed below.

Beware of characters arriving with parity set because the pattern matching considers all 8 bits of a byte unless the local parity is other than NONE and SET DISPLAY is 7-BITS. Arriving characters are modified by first removing the parity bit, if parity is other than NONE, then they are passed through the SET TRANSLATION INPUT converter, the high bit is again suppressed if SET DISPLAY is 7-BITs, the result is logged and stored for pattern matching.

The REINPUT command

Syntax: `REINPUT [`*timeout*`] {`*search-string*`, @`*filespec*`}`

The REINPUT command is like INPUT except that characters are read from the 128 byte serial port history buffer rather than always seeking fresh input from the port. The purpose is to permit the current text to be examined several times, looking for different match strings. A common case is reading the results of a connection message from a modem which might be "CONNECT 1200" or "CONNECT 2400", depending on the other modem. If the history buffer has less than 128 bytes then fresh input may be requested while seeking a match, until the buffer is full. REINPUT match searches begin at the start of the buffer whereas INPUT searches never go back over examined characters. REINPUT sets the testable status of SUCCESS or FAILURE and DOS ERRORLEVEL, just as for INPUT.

When a script fails because an INPUT or REINPUT command did not encounter the desired string within the timeout interval the message "?Timeout" is displayed.

The OUTPUT command

Syntax: `OUTPUT {`*string*`, @`*filespec*`}`

The OUTPUT command writes the indicated character string to the serial port as ordinary text. The string may contain control or other special binary characters by representing them in backslash form. Carriage Return `(CR)`, for example, is `\13 decimal`, `\o15` octal, or `\x0D` hexadecimal. The string may use 8-bit characters if the communications parity is type NONE. A special notation is also provided, `\b` or `\B`, which causes a BREAK signal to be transmitted.

The string to be transmitted starts with the first non-spacing character after the OUTPUT command and ends at either the end of line or, if executed within a TAKE file, at a semicolon (if you need to output a semicolon from within a TAKE file, use backslash notation, e.g. ‘‘`\59`’’). Indirectly obtained strings, the *@filespec* form, read the first line of the file up to but not including the explicit carriage return.

As a convenience, text arriving at the serial port during the OUTPUT command is shown on the screen if SET INPUT-ECHO is ON, and stored in a 128-byte internal buffer for rereading by subsequent (RE)INPUT commands.

The INPUT, REINPUT, and OUTPUT commands have a special syntax to replace the normal string with text obtained from a file or device:

```
OUTPUT @filespec
INPUT @filespec
```

Both forms read one line of text from the file or device and use it as the desired string. A common use is to wait for a password prompt and then read the password from the console keyboard. A string starts with the first non-spacing character and ends at either the end of line or, if executed within a TAKE file, at a semicolon. Indirectly obtained strings, the *@filespec* form, read the first line of the file up to but not including the explicit carriage return. Note if a trailing carriage return is needed it must be expressed numerically, such as `\13` decimal. Example:

```
input 7 Password:
echo Please type your password:
output @con
output \13
echo \13\10Thank you!
```

In this example, a TAKE file requests the user to type in the password interactively, so that it does not have to be stored on disk as part of the TAKE file.

The PAUSE command

Syntax: `PAUSE [{`*number*`, `*hh:mm:ss*`}]`

PAUSE turns on the DTR signal, and then waits one or more seconds, or until the specified time of day. Pauses are frequently necessary to avoid overdriving the host and to let a modem proceed through a dialing sequence without interruptions from Kermit. The default waiting time is set by SET INPUT DEFAULT-TIMEOUT and is normally one second. The optional integer number selects the number of seconds to pause for this command, and the *hh:mm:ss* selects a specific time of day. An explicit value of zero produces a pause of just a few milliseconds which can be useful in some situations.

Text arriving during the PAUSE interval is shown on the screen, if SET INPUT-ECHO is ON, and stored in a 128-byte internal buffer for rereading by a following INPUT command.

PAUSE is interrupted if there is any activity on the keyboard. Thus PAUSE can be useful for operations like:

```
echo "Type any key when ready..."
pause 9999
```

PAUSE is useful in scripts that are to be executed at some future time. For instance, if you want your PC to dial up another computer and transfer some files at 9:30pm, when the phone rates are lower, you can put the command

```
PAUSE 21:30:00
```

in your script file. Note that you cannot specify a time more than 12 hours in the future. If you need to pause until a specific time that is more than 12 hours away, you can use multiple PAUSE statements:

```
PAUSE 21:30:00  ; Pause until 9:30pm tonight
PAUSE  9:30:00  ; Pause until 9:30am tomorrow morning
```

Because PAUSE turns on the DTR signal, it can be useful in scripts where DTR must be asserted for a second or two to wake up the device your PC is connected to, before you can send any characters to it:

```
pause 1         ; Assert DTR and pause for 1 second
output \13      ; Send a carriage return
```

The WAIT Command

Syntax: `WAIT [{`*number*`, `*hh:mm:ss*`}] [\CD] [\CTS] [\DSR]`

WAIT performs a timed PAUSE, as above, but also examines the modem control signals Carrier Detect (`\CD`), Clear To Send (`\CTS`), and/or Data Set (modem) Ready (`\DSR`). If all of the signals specified in the WAIT statement are ON, or become ON before the timeout interval, the wait operation ceases with an indication of SUCCESS. If the time interval expires without all of the specified signals on, the status is FAILURE. Example:

```
Kermit-MS> wait 12:45:00 \cd \dsr
```

This waits until both CD and DSR asserted or until 45 minutes past noon, whichever happens first, returning SUCCESS or FAILURE respectively.

If no modem signals are specified, then WAIT is the same as PAUSE.

Labels and the GOTO Command

Labels and the GOTO command work together in the same fashion as in DOS Batch files. A label is a line which starts with a colon (:) in the leftmost column followed immediately by a word of text (no intervening spaces); material on the line after the label is ignored. The GOTO command is followed by a label, the leading colon is optional in the GOTO command. The label may be located either before or after the GOTO command and is found by searching the TAKE file or macro from the beginning. Thus, duplicated labels will always use the first occurrence. The target label must be in the current TAKE file or macro; one may not GOTO a label in another TAKE file or macro. Example:

```
:LOOP
echo again and\32
goto loop
```

will print "again and again and again and..." forever (until you type Ctrl-C). As a macro:

```
define test :loop,echo again and\32,goto loop
do test
```

Note that if a label follows a comma in a macro definition, there must be no intervening spaces:

```
define test ..., :top, ..., goto top  ; bad, space before colon.
define best ...,:top, ..., goto top   ; good, no space.
```

In this example, the best macro will work, the test macro won't.

The IF Command

Syntax: `IF` *test-condition MS-Kermit Command*

The IF command gives MS-Kermit scripts the ability to make a decision based upon the criterion specified as the *test-condition*. If the test condition is true, then the command is executed. Otherwise, it is skipped. The test conditions are:

NOT Modifier for other conditions below.

ALARM True if the current time of day is at or later than the alarm clock time. The alarm clock time is set by the command SET ALARM time. IF ALARM distinguishes early from late with a 12 hour field of view.

COUNT True if the current COUNT variable is greater than zero. COUNT is a special Kermit variable for each active TAKE file or macro. It is set by the command SET COUNT and it is both tested and modified by the IF COUNT command. The intent is to construct simple script loops where the IF COUNT command first decreases COUNT by one (but never below zero) and then if COUNT is greater than zero the following Kermit command is executed. Because COUNT exists only for TAKE files and macros it cannot be used interactively. Each TAKE file or macro has its own distinct copy of COUNT, and nested TAKE files or macros do not interact through their COUNTs. Initially COUNT is zero.

DEFINED *symbol*
True if the named macro or variable is defined. You can use this feature to remember things for future reference.

EQUAL *word1 word2 command*
True if the two words are lexically equal. Alphabetic case is ignored unless SET INPUT CASE OBSERVE. If they match, the following command is executed. The modifier NOT may be inserted to invert the sense of the test. Substitution variables may be used in place of *word1* and *word2,* but the command will only work if these variables contain single words, not phrases. If *word1* or *word2* begin with `@`, then the rest of the word is interpreted as a file specification, and the first word in the file is used.

ERRORLEVEL *number*
True if the DOS errorlevel number matches or exceeds the given (decimal) number.

EXIST *filespec*
True if the specified file exists.

FAILURE
True if the previous status-returning Kermit command reported failure.

SUCCESS
True if the previous status-returning Kermit command reported success. When using IF SUCCESS and IF FAILURE, it is important to SET INPUT TIMEOUT PROCEED, otherwise the script will quit immediately upon a failing INPUT or REINPUT, before getting to the IF statement.

IF commands are closely modeled on those of DOS Batch files, for familiarity. They consist of a test condition, perhaps modified by the leading word NOT, and then any legal Kermit command. GOTO is an especially useful command here to branch in the TAKE file or macro.

The "object" of an IF command is a Kermit command, which can be:

- A regular, predefined Kermit command, like `SEND FOO.BAR` or `SET SPEED 1200`.
- A GOTO, allowing subsequent statements to be skipped.
- Another IF command, as in `IF DEFINED \%3 IF EXIST FOO.BAR SEND FOO.BAR`. The SEND command is executed only if both IF conditions are true.
- A macro. This allows a semblence of structured programming, with an implied "begin" and "end" around the commands that compose the macro. For instance:

```
define giveup echo I give up!, hangup, stop
input 10 Login:
if failure giveup
output myusername
```

The Kermit commands which yield SUCCESS or FAILURE conditions are: GET, SEND, RECEIVE, the REMOTE commands, INPUT, REINPUT, BYE, FINISH, LOGOUT, and WAIT.

The POP and STOP Commands

Use these commands for terminating execution of a TAKE file or macro. POP terminates the current level and returns to the previous level. For example, if you gave the command "take shower", and the SHOWER file contained a command "take bath", and the BATH file contained a command "take hike", and a POP command was encountered in the HIKE file, then the next command executed would be the one following the "take hike" command in the BATH file. If a STOP command was encountered in any of these files, MS-Kermit would return immediately to interactive command level. POP and STOP work in similar fashion with nested macro invocations: POP returns to the invoking macro, STOP returns to command level.

Script Examples

A counting loop. This TAKE file excerpt says hello three times, then says goodbye:

```
set count 3             ; Prime the loop counter for three passes
:TOP                    ; A label for GOTO
echo Hello\13           ; Something to see, with carriage return
if count goto top       ; Loop if COUNT is greater than zero
echo Goodbye!\13
```

Figure 1-2 shows a simple script file that logs in to a computer, prompting the user for her password using the `@con` construction, and then connects as a terminal. Notice the semicolons used to indicate comments in TAKE files. If these same commands were typed by hand at the Kermit prompt the semicolon material would be considered part of a string! Typing a Control-C will interrupt and terminate any of the commands.

Figure 1-3 illustrates some detailed control of the Hayes 2400 modem. Some understanding of the Hayes dialing language is helpful for deciphering this script (consult your Hayes modem manual). If the script is stored in a file called `HAYES.SCR`, then a DIAL macro can be defined like this:

```
define ermsg echo %\1\13, stop       ; Define an error handling macro.
clear                                ; Clear the input buffer.
set speed 9600                       ; Set the transmission speed.
output \13                           ; Carriage return to awaken host.
input 15 Login:                      ; Wait up to 15 secs for prompt.
if failure ermsg No_login_prompt!    ; Give up if none.
output Sari\13                       ; Send username and CR.
set input echo off                   ; Privacy, please.
input 5 Password:                    ; Quietly wait for this.
if failure ermsg No_password_prompt! ; Give up if it doesn't come.
echo Type your password now...       ; Make our own prompt.
output @CON                          ; Send console keystrokes.
output \13                           ; Add a real carriage return.
input 30 $                           ; Wait for system prompt.
if failure ermsg No_system_prompt!   ; Give up if none.
connect                              ; Start terminal emulation.
```

Figure 1-2: MS-Kermit Script for Logging In

```
define dial take hayes.scr
```

The trick here is that any invocation of the "dial" or "do dial" command with an operand will set the variable `\%1`, which is used in the TAKE file, for instance:

```
dial 765-4321
```

will set `\%1` to "765-4321", the number to be dialed. You can also type

```
dial {212 765 4321}
```

if you want to include spaces in the phone number. This script requires version 2.32 of Kermit or later.

A combination of DOS Batch and Kermit Script files is shown in Figures 1-4 and 1-5 (see your DOS manual for an explanation of the batch file syntax). The purpose is to allow a user to say ‘‘`SEND` *filename*’’ at the DOS prompt. The DOS batch shell, `SEND.BAT`, and the login script, `KX`, are combined to login to a VAX through a data switch, run VMS Kermit in server mode, transfer the file, submit it to VMS Mail, delete the disk file, shut down the server and logout from the VAX, and report the overall transfer status. The user is asked to provide a password interactively.

1.9. Initialization Files Revisited

At Columbia University, we have IBM 370-series mainframes running VM/CMS, and VAX and SUN systems running Unix. All of these systems are accessible through an IBM/Rolm (now Siemens/Rolm) voice/data CBX. The IBM systems have two different kinds of front ends, a COMTEN 3695 (similar to IBM 3705) for linemode half-duplex connections, and various Series/1-style protocol converters (including the 7171 and 4994) for full-screen, full-duplex 3270 emulation, all of which use various combinations of parity and other settings. The VAX is connected directly to the CBX, whereas the SUNs are connected to the CBX through Cisco Ethernet terminal servers. Figure 1-6 shows the `MSKERMIT.INI` file used at Columbia for automatic login to these systems. It illustrates the creative use of macros and scripts. Numerous site- and system-dependent key definitions have been omitted.

A bit of explanation might clarify some of this. The IBM/Rolm CBX prompt is "CALL, DISPLAY OR MODIFY?" and we respond with a CALL command for the desired system or front end, like CALL SIMB (IBM mainframe in full screen mode through a 7171 protocol converter), CALL CUVMB (IBM mainframe in linemode through the COMTEN), CALL CUNIXC (a VAX), or CALL CUNIXA (a SUN, through an Ethernet terminal server). When the initial call through the CBX is completed, the message "CALL COMPLETE" appears, and then begins the

```
def errstop echo \%1\13, def \%1, hang, stop ; Error handler.
if not defined \%1 errstop {Please supply a phone number!}
assign \%n \%1                       ; Copy the phone number.
clear                                ; Clear the input buffer.
set speed 2400                       ; Dial at high speed.
wait 2 \cts                          ; Is modem turned on?
if fail errstop {Please turn on your modem.} ; No.
echo Initializing modem...\13\10     ; Yes.
output ATZ\13                        ; Reset the modem.
pause 2                              ; Give it a little time.
output AT F1 Q0 V1 X4 S0=0\13        ; Put modem in known state.
input 8 OK                           ; Look for response.
if fail errstop {Can't initialize modem.}
pause 1                              ; Pause for a second first.
set count 5                          ; Set the redial limit.
define \%d \13Dialing                ; Initial dial message.
:REDIAL
echo \%d \%n...\13\10                ; Tell them we're dialing.
output ATDT\%n\13                    ; Dial the phone number.
clear                                ; Clear away the command echo.
input 60 CONNECT                     ; Wait for CONNECT message.
if success goto speed                ; Got it, go check speed.
define \%m No dialtone or no answer. ; Make this the error message.
reinput BUSY                         ; Didn't connect.  Was it busy?
if failure errstop {\%m\10\13Try again later.} ; No
Echo \13Busy...                      ; It's busy, let them know.
hangup                               ; Drop DTR momentarily.
pause 60                             ; Wait one minute.
define \%d \13Redialing              ; Change message to "Redialing".
if count goto redial                 ; Then go redial.
define \%m \13Line busy.             ; After 5 tries set this message.
:SPEED                               ; Connected!
pause 1                              ; Wait for text after CONNECT.
define \%s 2400                      ; Assume speed is 2400.
reinput 1 2400                       ; Rescan current text for "2400"
if success goto done                 ; It is.
define \%s 1200                      ; It isn't, so assume 1200.
reinput 1 1200                       ; Is it?
if failure define \%s 300            ; It isn't, so it must be 300.
:DONE                                ; We know the speed.
set speed \%s                        ; So set it.
echo Connecting at \%s bps...\13     ; Tell the user.
pause 2                              ; Give her a chance to read it.
set terminal clear                   ; Clear screen.
define \%1                           ; Clear argument.
connect                              ; And start terminal emulation.
```

Figure 1-3: MS-Kermit Script for More Control of a Hayes 2400 bps Modem

interaction with the desired host, front end, or terminal server, each of which has its own set of prompts and responses.

To connect to a given system, one types "do simb", "do cunixc" to invoke a "connecting" macro. Each of these, in turn, invokes the CBX macro to navigate through the CBX to the desired system. If the CALL COMPLETE message is encountered, then further macros (3695, 7171, etc) are used to get past any associated front end (e.g. to tell the COMTEN which IBM mainframe is wanted, or to tell the protocol converter what terminal to emulate), and then to login on the desired system, prompting on the screen for user ID and password. Finally, a macro like "vml" (VM linemode), "xed" (XEDIT, i.e. VM full screen), or "dec" (VAX or SUN) is executed to set the communication parameters for the system just logged in to. The key definitions that are shown in the "vml", "xed", and "dec" macros assign the host's character deletion code (backspace or rubout) to the AT's backarrow key.

File SEND.BAT, DOS batch program:

```
echo off
Rem Kermit, one-line file mailer, by Joe Doupnik.
Rem Logon to VAX, run Kermit, Send user’s file,
Rem  post via MAIL, logout from VAX.
if ".%2" == "." goto usage
if exist %1 goto proceed
echo No file to send!
:usage
echo Usage is SEND filename username
goto done
:proceed
echo Logging onto the Vax ...
kermit set disp q,take kx,send %1,pau,rem host mail %1 %2,pau 2,bye,
if errorlevel 3 goto badrem
if errorlevel 2 goto badrcv
if errorlevel 1 goto badsnd
echo File(s) "%1" has been mailed to %2.
goto done
:badrem
echo Mail did not cooperate!
:badrcv
echo Receive failed!
goto done
:badsnd
echo Send failed!
goto done
:done
echo on
```

Figure 1-4: MS-DOS Batch File Invoking Kermit to Send VAX Mail

1.10. International Character Sets

MS-Kermit may be used on the IBM family and compatibles for interacting with host computers in different languages. MS Kermit CONNECT mode has separate translation mechanisms for screen and keyboard. Keyboard translations are managed through the SET KEY facility which maintains a table of defined keys and their output values (single characters, strings, or Kermit keyboard verbs). The keyboard is normally read via the system Bios, but it may also be read via DOS (with a loss of some key combinations) by saying SET KEY OFF (i.e., turn off Bios reading). The keyboard can be modified rapidly by a group of SET KEY commands placed in a macro. The host has no direct control of the keyboard translations; the host thinks Kermit is a real Digital VT102/VT52 or Tektronix 4010 terminal and those devices do not have redefinable keys.

Screen translation is accomplished in two places, the SET TRANSLATION INPUT table and the built-in character sets. SET TRANSLATION INPUT is a table of received versus reported character codes, and it is enabled by SET TRANSLATION INPUT ON. The table is initially an identity which allows individual entries to be modified as desired by the command

SET TRANSLATION INPUT <*received-code*> <*displayed-code*>

Only characters destined for the screen as text or cursor control (control codes) are translated; escape sequences and transparent printing characters bypass the SET TRANSLATION table. The table is bypassed for printing to permit binary graphics streams to be sent to the printer. A character about to be shown on the screen can be modified by selection of a character set, such as US-ASCII, UK-ASCII, ALTERNATE-ROM, or line drawing.

File KX, Kermit script:

```
Comment Login script for VAXA via Micom data PBX Switch.
set input timeout quit
set input echo off
set display quiet
output \13
comment - "slowly." and "CLASS" are part of the switch's prompt.
input 10 slowly.
input 10 CLASS
pause
comment - Slowly tell switch "vaxa", wait for beep.
output v
output a
output x
output a
output \13
pause
input 5 \7
comment - Done with Switch, wake up the VAX and log in.
pause
output \13
pause
input 5 Username:
set input timeout proceed
output MYNAME\13
input 2 Password:
comment - Prompt ourselves, then get password from console.
echo Enter password:
output @con
comment - Send a carriage return at the end of the password.
output \13
comment - Expect ESC Z from the VAX's Set Term/Inquire...
comment - Respond ESC [ <query symbol> 6 c   (say we are VT102).
comment - Note syntax for including question mark!
input 15 \27Z
output \27[\{63}6c
comment Look for VMS dollar sign prompt
input 15 $
comment Start VMS Kermit and place it in server mode
output kermit server\13
comment - allow server's message to finish, "machine." appears twice.
input 10 machine.
input 10 machine.
pause
```

Figure 1-5: MS-Kermit Script for Logging into VAX and Sending Mail

The SET TRANSLATION INPUT mechanism operates at the Kermit command level and is available to macros, TAKE files, and hand typed control. Host control is available only indirectly via the special macros TERMINALR and TERMINALS, discussed below, which may contain the SET TRANSLATION INPUT and other commands.

Character sets can be selected either by the Kermit command SET TERMINAL CHARACTER-SET (expressed by hand, in macros, or in Take files), or by host control of the terminal emulator via the escape sequences `ESC (` *char* or `ESC )` *char* and the Control-O and Control-N codes. Thus, rapid changes of displayed characters is available to the host and to the user through all three dynamic pathways: macros, Take files, hand typing or received codes.

Version 2.32 of MS-Kermit also includes a new ability to operate right-to-left during CONNECT mode, in order to

```
; MS-Kermit 2.31, 2.32 Initialization File for the IBM PC, XT, AT, PS2, etc.
; Christine Gianone, Vace Kundakci, Columbia University, December 1988
echo Columbia University IBM PC Kermit Initialization file...

; User IDs on various systems.  Substitute your own IDs.
def \%c XYZCU          ; User ID for IBM mainframe
def \%u xyz            ; UNIX ID for UNIX

; General settings
set warning on         ; Change this to "off" to allow overwriting of files.
set speed 9600         ; Use 9600 bits per second by default
set term vt102         ; Emulate a DEC VT-102 terminal
set term wrap on       ; Have Kermit wrap lines at column 80

; Behavior of INPUT command in script programs
set input timeout quit ; Exit from script if input pattern not found
set input echo on      ; Echo characters that arrive during INPUT
set input case observe ; Match according to alphabetic case

; Macros for connecting to different systems thru the IBM/Rolm CBX
def cuvmb  do cbx,o c cuvm\13,  i 10 PLETE,      do 3695, o vmb\13, do 4381
def simb   do cbx,o c simb\13,  i 10 PLETE, pau, do 7171, do 3270
def cunix  do cbx,o c cunix\13, i 10 PLETE, pau, do cuts, do unix
def cunixa def \%s cunixa,do cunix
def cunixb def \%s cunixb,do cunix
def cunixc do cbx,o c cunixc\13,i 10 PLETE, pau, out \13, do unix

; Macros for navigating thru front end and login prompts
def cbx  do def,o \13,i 10 MODIFY?                          ; IBM/Rolm CBX
def 3695 i 5 ING CHARACTERS:\32\32                          ; COMTEN
def 7171 pau,cle,o \13,i 5 TERMINAL TYPE:\32,o vt-100\13    ; 7171 front end
def 4381 do vml,i 5 BREAK KEY,o \b,i 5 .\17,o LOG \%c\13,c  ; VM/CMS linemode
def 3270 pau,cle,o \13,o L \%c\13,do vmf,c                  ; VM/CMS fullsc.

; CU Terminal Servers (cutsa, cutsb, etc)
def cuts set inp tim p,out \13,pau,set co 8,:loop,out \13,i 3 >,-
if suc goto ok,if cou goto loop,ech Failed,stop,:ok,out \%s\13,set inp tim q

; UNIX login with speed matching
def unix set inp timeout proc,set count 8,-
:loop,i 5 login:\32,if suc goto ok,out \13,if count goto loop,-
echo Failed,stop,:ok,out \%u\13,do dec,set inp tim q,connect

; Macros for interacting with different systems:
def vml do tty,set par m,set k \270 \8,  set k \3 \Kbreak    ; VM linemode
def vmf do def,set par e,set k \270 \8,  set k \3 \3,do simk ; VM fullscreen
def dec do def,set par n,set k \270 \127,set k \3 \3         ; DEC, SUN, etc
def def set tim of,set loc of,set hand non,set flow xon,do nosimk ; Default
def tty set tim on,set loc on,set hand xon,set flow non,do nosimk ; IBM TTY
```

Figure 1-6: An Advanced MS-Kermit Initialization File

interact with Hebrew or Arabic language applications on the host computer. The pertinent commands are SET TERMINAL DIRECTION {LEFT-TO-RIGHT | RIGHT-TO-LEFT}, and SET TERMINAL CHARACTER-SET ALTERNATE-ROM. The latter command makes these high bit characters available by active user selection, or by reception of the escape sequences below to associate them with one of the two VT102 character set pointers called G0 (normal) and G1 (alternate). Arrival of Control-O selects the G0 set (default) and Control-N the G1 set.

In addition, two special macro names TERMINALR and TERMINALS have been set aside, which can be invoked within the VT102 emulator by reception from the host of the special escape sequences:

`ESC [ ? 34 h` (invokes macro TERMINALS)
`ESC [ ? 34 l` (lower case L, invokes macro TERMINALR)

and/or by using new keyboard "verbs" (not preassigned to keys):

`\Kterminals` (invokes macro TERMINALS)
`\Kterminalr` (invokes macro TERMINALR)

When these macros are invoked within the terminal emulator and if they are defined then CONNECT mode is exited and the macro is executed. There is no automatic return to Connect mode at the completion of the macro. If the macro is not defined then CONNECT is not exited and nothing happens. Initially neither macro is defined. If a return to Connect mode is desired then include CONNECT in the macro. Any legal action is permitted in these macros, including invoking other macros and Take files.

The purpose of these two names and macros is to allow a host or the local user to interactively select two local operations while within the terminal emulator, such as changing language specific setups or other desirable things, which are much more involved than an existing keyboard verb. There is no restriction on what the macros may do since Kermit is then operating not in Connect mode but at the Kermit command prompt level, as it is for other macros.

The escape sequences above are a Kermit specific extensions of Digital Equipment Corporation’s private escape sequences to set and reset modes; hence the letters S and R in the macro names.

One suggestion for employing SET TERM DIRECTION, SET TERM CHARACTER, and the macros TERMINALR and TERMINALS to facilitate mixed Hebrew and English communications is the simple Take file below:

```
; Define macros hebrew and english to do all the work

def hebrew  set term dir right, set term char alt, hkey, comkey
def english set term dir left, set term char us, set key clear, comkey

; Define host-reachable macros for on the fly changes while
; staying in the emulator

def terminalr english, connect
def terminals hebrew, connect

; Define IBM-PC F1 key as switch to English, F2 as switch to Hebrew.
; Done here to be remembered despite SET KEY CLEAR in macro English.
; F1 and F2 thus are user-level commands during emulation.

def comkey set k \315 \Kterminalr,set k \316 \Kterminals

; Define SET KEYs for Hebrew keyboard layout via macro hkey

def hkey set k \x27 \x2c,set k \x2c \x9a,set k . \x95,set k / \x2e,-
 set k \x3b \x93,set k \x60 \x3b,set k a \x99,set k b \x90, hkey1
def hkey1 set k c \x81,set k d \x82,set k e \x97,set k f \x8b,-
 set k g \x92,set k h \x89,set k i \x8f,set k k \x87,hkey2
def hkey2 set k l \x8c,set k m \x8a,set k m \x96,set k n \x8e,-
 set k o \x8d,set k p \x94,set k q /,set k r \x98,set k s \x83,hkey3
def hkey3 set k t \x80,set k u \x85,set k v \x84,set k x \x91,-
 set k y \x88,set k z \x86
```

After executing this file, one may switch Connect mode language support between Hebrew (right to left, national display characters, similarly translate outgoing keyboard characters) and English by stating a single keyword at the Kermit prompt, "Hebrew" or "English", or while within Connect mode by pushing the F1 or F2 keys (in this example), or by reception of `ESC [ ? 34 h` or `l` from the host. All the work is done from memory material and is essentially instantaneous. Clearly, other languages can also utilize these tools.

IBM PCs or compatibles will normally have national characters installed in the upper portion of the character set ROM, in positions 80H-9AH. EGA systems generally come with a program to load the appropriate national character set into this portion of memory, such as HEBEGA for Hebrew. Version 3.30 (and later) of DOS supports the notion of "Code Page" for PS/2 systems, or other systems with EGA or LCD adapters, described in Appendices B and C of the DOS 3.30 reference manual.

1.11. MS-Kermit Features for Different Systems

As noted early on, MS-Kermit was designed primarily for the IBM PC family, and later adapted to various non-IBM-compatible MS-DOS (and even non-MS-DOS) systems. Some of these adaptations provide all the features of the IBM PC version, others provide only a subset, and still others may include features not available on the IBM family. These features are all of the system-dependent variety; the Kermit file transfer protocol should be implemented identically on all versions of MS-Kermit. The most obvious differences are in the terminal emulation options and the keyboards. Table 1-7 shows the terminal emulation options for the systems presently supported by Kermit-MS, and Table 1-8, shows which keys are used for screen rollback on the various systems supported by MS-Kermit.

System	EscChar	Capabilities	Terminal Service
ACT Apricot	^]	K	VT52 ???
DEC Rainbow	^]	R P K D	VT102 firmware
DECmate/DOS	^]	K	VT100
Generic DOS	^]	K	Depends on system
Grid Compass	^]	K	???
HP-110	^]	K	Dumb terminal
HP-150	^]	R K	HP-2623 firmware
IBM PC family	^]	R M P K D	H19,VT52,VT102,Tek emulation
Intel 3xx	^]	K	Uses real terminal
NEC 9801	^]	M P K D	VT102, Tektronix emulation
NEC APC3	^]	R M P K D	H19,VT52,VT102 emulation
NEC APC	^]	R P K	VT100, ADM3A firmware
Olivetti M24	^]	R M P K D	Same as IBM PC
Sanyo MBC55x	^]	R M P K D	H19,VT52,VT102 emulation
Wang PC	^A	K	Wang firmware
TI Pro	^]	M P K	VT100/Tektronix
Victor 9000	Alt-]	M P K D	H19,VT52,VT102 and/or Tek4010
Zenith Z100	^]	K	Heath-19 emulation

R=Rollback, M=Modeline, P=Printer control, K=Key redefinition, D=screen Dump

Table 1-7: Kermit-MS Terminal Emulation Options

Another difference is the default communication port, the number of communication ports supported, and the names given to them. For instance, the IBM PC family supports COM1 and COM2, and uses COM1 by default. MS-Kermit may be persuaded to support higher-numbered IBM ports using the method outlined in section 1.18.3. For remote operation, IBM's name for the console is CON, so if you CTTY COM1, you do CTTY CON to put the PC back to normal.

System	Screen Down	Line Down	Screen Up	Line Up
IBM PC	PgUp	Ctrl-PgUp	PgDn	Ctrl-PgDn
Rainbow	PrevScreen	Ctrl-PrevScreen	NextScreen	Ctrl-NextScreen
HP-150	Prev	Shift-UpArrow	Next	Shift-DownArrow
NEC APC	Uparrow	Ctrl-UpArrow	DownArrow	Ctrl-DownArrow
NEC APC3	PgUp	Ctrl-PgUp	PgDn	Ctrl-PgDn
Sanyo 55x	PgUp	Ctrl-RtArrow	PgDn	Ctrl-PgDn

The IBM PC also allows use of the Home key to get to the top of its display memory and End key to get to the bottom, and the keypad minus (`-`) key to toggle the mode line on and off. The Rainbow uses Shift-Next-Screen to get to the bottom of its display memory, but provides no key for moving directly to the top.

Table 1-8: Kermit-MS Screen Scroll Keys

The DEC Rainbow

The DEC Rainbow version of MS-Kermit uses the built-in VT102 terminal firmware and setup modes, and can operate at speeds up to 9600 baud. It has no 25th screen line, and therefore no Kermit mode line during CONNECT. It supports only the Rainbow's single communication port, and not the printer port, so SET PORT for the Rainbow is not implemented (but of course the printer may be used for printing.) The Rainbow may be put in remote mode by CTTY AUX, and returned to normal with CTTY SCRN. The Rainbow supports several SET TERMINAL commands: VT102, VT52, and ROLL.

The keypad and cursor keys all work properly in VT102 and VT52 modes and in application as well as native states (they never had in previous versions). Newline mode is activated for received characters (LF ==> CR/LF). Screen roll back is almost 11 screenfuls. Table 1-9 shows the verb names and default key assignments for the Rainbow. On the main typewriter keyboard the shifted comma and period are converted to special keys available for Set Key assignment without impacting the normal unshifted ASCII actions; Shift Lock has no effect on these keys.

The DECmate II

MS-Kermit for the DECmate II with the XPU option is somewhat similar to Rainbow Kermit. It uses built-in terminal VT100 firmware and setup modes and baud rates up to 9600 on the single communication port. The printer port is not available for communications in this version. There is no mode line, but other connect-mode escapes are supported, including sending BREAK. Disks A through I are supported, and the floppy disk format is compatible with the Rainbow. DEC utilities are available for file conversion between DOS and WPS-8 files.

The NEC APC3

The NEC APC3 version of MS-Kermit assumes that the `ANSI.SYS` driver has been installed and that a color monitor is being used; the color graphics option is not used by Kermit. Although the display should be entirely sensible with a monochrome system, it has not been tested. Differences from the IBM PC version include:

SET BAUD: The useful baud rates supported range from 300 to 9600.

SET PORT: The available ports are 1, 2, 3, or their equivalents AUX, AUX2, AUX3.

SET TERMINAL COLOR: Instead of specifying colors by number, the words BLUE, RED, MAGENTA, GREEN, CYAN, YELLOW, or WHITE are appropriate. This is the color of the text in connect mode; background colors are not available. Monochrome monitors will respond with display changing from most dim to most bright if the colors are specified in the order given.

SET TERMINAL KEYCLICK: Not implemented in Kermit; use the NEC provided command.

Rainbow Key	Verb Name	Operation
PF1	\Kpf1,\Kgold	Keypad function key
PF2..PF4	\Kpf2..\Kpf4	Keypad function keys
keypad 0..9	\Kkp0..\Kkp9	Keypad digit keys
keypad -	\Kkpminus	Keypad minus key
keypad ,	\Kkpcoma	Keypad commma
keypad .	\Kkpdot	Keypad dot (period) key
keypad Enter	\Kkpenter	Keypad Enter key
up arrow	\Kuparr	Cursor keys
down arrow	\Kdnarr	
left arrow	\Klfarr	
right arrow	\Krtarr	
Shift Prev Screen	\Khome	Rewind to start of screen buffer
Shift Next Screen	\Kend	Unwind to end of screen buffer
Ctrl Prev screen	\Kupone	Backup one screen line
Ctrl Next screen	\Kdnone	Advance one screen line
Prev screen	\Kupscn	Backup one screen
Next screen	\Kdnscn	Advance one screen
Print Screen	\Kprtscr	Copy screen to printer
Ctrl Print Screen	\Ktoggle_prn	Toggle echoing screen to printer (printer failure resets toggle)
Do	\Kdump	Copy screen to file (KERMIT.SCN)
Break	\Kbreak	Send a BREAK
Shift Break	\Klbreak	Send a Long BREAK
Main Screen	\KDOS	Push to DOS
Help	\Khelp	Show Connect mode help menu
Exit	\Kexit	Exit Connect mode
*	\Knull	send a null out the serial port
*	\Khangup	hangup phone by dropping DTR, RTS
*	\Klogon	resume logging, if active
*	\Klogof	suspend logging
*	\Kstatus	display status table

* (verbs not pre-assigned to keys)

Table 1-9: Kermit-MS Verbs for the DEC Rainbow

SET TERMINAL SCREEN-BACKGROUND: Not implemented.

During terminal emulation, screen scroll is handled by the PgUp and PgDn keys. If used in combination with the Ctrl key, the display moves but one line. If used in combination with the Fnc key, the display scrolls to the end of the buffer. The Fnc-INS combination toggles the mode line on/off. The Fnc-DEL combination toggles the terminal emulation type. The Fnc-Break combination resets the emulator. The Help key pulls down the connect mode menu. The ANSI escape sequence for disable/enable cursor is implemented.

1.12. Compatibility with Older Versions of MS-DOS Kermit

The last monolithic (single source file) release of MS-DOS Kermit was 1.20. Meanwhile, implementations based on versions of that vintage will have at least the following incompatibilies from the version described here:

- "RECEIVE filespec" is used instead of "GET filespec". There is no GET command in older versions, and no way to specify a new name for an incoming file.
- No LOCAL or REMOTE commands.
- No 8th-bit prefixing, repeat counts, CRCs or 2-character checksums.
- No TAKE or initialization files.
- No command macros or command line arguments.
- No terminal session logging.

and others, depending on the specific version.

Incompatibilities between 2.29 and later releases include:

- LOCAL command has been removed from 2.30 and later.
- CLEAR command now means clear serial port buffer rather than key and macro definitions. Key and macro definition string space is now garbage collected, so a CLEAR command for them is no longer necessary.
- CLRINP command is gone (replaced by CLEAR).
- Numbers of the form *nnn* default to decimal rather than octal.
- Status of Default Disk is now shown as default disk and path.
- LOG *filespec* replaced by LOG SESSION *filespec* and LOG PACKET *filespec*.
- SET KEY and SHOW KEY commands use different key identifications and syntax:

MS-Kermit no longer understands keycap names such as F1 and BACKSPACE because the codes are now highly dependent on individual keyboards, software, and computers. Also, not every key press combination is supported by the system software and key codes do depend on the keyboard in use. Thus, the SHOW KEY command is normally used to obtain codes for keys on your system. In most cases, defining one key also redefines all other keys sending the same character. This is a side effect of not knowing the physical details of every keyboard. However, efforts have been made to recognize many such "aliased" keys and to generate unique identifications for each. Special keys, such as F1, F2 and others which do not send an ASCII code are usually unique and are identified by scan codes.

Previous versions of MS Kermit used a different key coding algorithm and not all old codes map to the expected keys. However, Kermit does attempt to use the older SET KEY syntax properly as much as possible. The older syntax required the keyword SCAN followed by a number WITHOUT the BACKSLASH. The current MS Kermit uses decimal as the default number base and previous versions used octal in certain commands. So, when Kermit senses an old style SET KEY command it converts the number, displays the new format and gives a warning message. It is best to make a new style SET KEY file.

1.13. What's Missing

Kermit-MS has plenty of room for improvement. Missing features (which may be added in future releases) include:

- Sliding window transport protocol.
- Default filetype for TAKE command files.
- Passing paramaters in TAKE command, like in DO command.
- A way to send files with their full path names.
- A way to play back session logs directly from disk to screen.
- Trapping of carrier loss during CONNECT or file transfer.
- A better built-in help facility.
- A way to dump or print Tektronix graphics screens.

1.14. Installation of Kermit-MS

If you already have Kermit on your PC, you can use it to obtain new versions of Kermit-MS when they appear on the central system at your site. If you do not have Kermit or any other reliable file capture facility on your PC, you can order a Kermit diskette from Columbia (write to Kermit Distribution, Columbia University Center for Computing Activities, 612 West 115th Street, New York, NY 10025, USA, for information), or from any of a number of user groups or diskette services. If you don't have Kermit already, and absolutely can't get a Kermit diskette, but have access to another computer that has a copy of the MS-DOS Kermit program (usually in ``.BOO'' format, explained below), there are two recommended methods for getting it onto your PC:

1. Use another file capture facility to get it.
2. Type in and run the "baby Kermit" program (72 lines) from chapter 7 of the Kermit book.

The first method involves either "raw capture" (no error checking), or else use of another protocol, such as Xmodem, which, like Kermit, requires a program to execute the same protocol on both ends of the connection.

Raw capture generally involves "typing" the file on the other computer, with your PC taking the place of the terminal, and rather than displaying the file on the screen as it's being typed, your PC is storing it on the disk. This is a tricky process, however, because data can easily be lost or corrupted. For instance, you could write a very short BASIC program to capture a file in this way, but it could probably not keep up -- even at low baud rates -- with the transmission speed unless you included the tricky serial port BASIC commands. The DOS command COPY COM1 *filename* command has the same speed problem, and it stops only when it receives a Control-Z character from the other computer.

If the other computer has Kermit on it -- which is likely, since this is probably the reason you want to get Kermit onto your PC -- you should type in the receive-only BASIC Kermit program listed on pp.186-188 of the Kermit book, and then use it in conjunction with the other computer's Kermit to transfer the file. Make sure to set a long enough delay on the other computer to give yourself time to escape back to the PC and start up the "baby Kermit" before packets start to arrive, otherwise you'll probably get fatal DOS i/o errors.

Note that Kermit programs are often distributed under names other than "Kermit". The Columbia Kermit program library contains hundreds of Kermit programs, which must be given unique names. MS-DOS Kermit for the IBM PC, for instance, is called `MSVIBM`. Once you have this program in `.EXE` format on your disk, you probably should rename it to `KERMIT.EXE`, because the distribution name is harder to remember (and type).

You will probably also want to create an MS-Kermit initialization file. A sample is distributed with MS-Kermit as `MSVIBM.INI`. This should be tailored to your requirements, and then renamed to `MSKERMIT.INI`, and stored where Kermit can find it (in the current directory or any directory in your DOS PATH).

".BOO Files"

MS-Kermit (and many other Kermit programs) are often distributed using a special encoding called "boo" (short for "bootstrap") format, developed especially for distribution of MS-Kermit over networks and communication lines. MS-Kermit has grown to have so many features that the binary program image (the `.EXE` file) has become quite large. But binary files are generally not compatible with the common labeled tape formats (e.g. ANSI D), electronic mail, or raw downloading -- the methods most commonly used for Kermit distribution.

A common practice is to encode `.EXE` and other binary files into printable characters, such as hexadecimal digits, for transportability. A simple "hex" encoding results in two characters per 8-bit binary byte, plus CRLFs added every 80 (or less) hex characters to allow the file to pass through card-oriented links. A hex file is therefore more than twice as large as the original binary file.

A `.BOO` file is a more compact, but somewhat more complicated, encoding. Every three binary bytes (24 bits) are split up into four 6-bit bytes with 48 (ASCII character ``0'') added to each, resulting in four ASCII characters ranging from ``0'' (ASCII 48) to ``o'' (ASCII 111), with CRLFs added at or near "column 76". The resulting file size would therefore be about 4/3 the `.EXE` file size. This is still quite large, so `.BOO` files also compress consecutive null (zero) bytes. Up to 78 consecutive nulls are compressed into two characters. Tilde (‘‘~’’) is the null-compression lead-in, and the following character indicates how many nulls are represented (subtract 48 from this character’s ASCII value). For instance ‘‘`~A`’’ means 17 consecutive nulls; ‘‘~~’’ means 78 of them. Repeated nulls are very common in `.EXE` files.

4-for-3 encoding combined with null compression reduces the size of the encoded file to approximately the same size as the original `.EXE file`, and sometimes even smaller. The first line of a `.BOO` file is the name (in plain text) of the original file. Here’s what the first few lines of a typical `.BOO` file look like:

```
MSVIBM.EXE
CEYP0Id05@0P~3oomo2Y01FWeP8@007P000040HB4001‘W~28bL005\W~2JBP00722V0ZHPYP:
\8:H2]R2V0[‘PYP:68>H2S23V0YHPiP:Xg800;Qd~2UWD006Yg~2Ogl009]o~2L8000;20~~~~
~~~~~~~:R2H008TV?P761T410<H6@P40j4l6RRH0083l17@PP?‘1M@?YSP20o0Ee0nUD0h3l
1WD3jO@3]0VjW03=8L?X4‘N0o01h1\H6~20l>0i7n0o1]e7[@2\PO=8LH60@00Raj>04^97Xh0
```

Programs for Handling .BOO Files

Kermit Distribution includes several useful `.BOO`-file programs:

`MSBPCT.BAS` This Microsoft BASIC program can be used on any PC that has BASIC to decode a `.BOO` file into an `.EXE` file. It’s about 50 lines line, so it can be typed in.

`MSBPCT.BOO` BASIC programs run rather slowly, so `.BOO`-file decoders have also been written in high-level languages like C. The `MSBPCT.EXE` file that was produced by compiling `MSBPCT.C` is encoded into `MSBPCT.BOO`, which can be decoded back into `MSBPCT.EXE` using `MSBPCT.BAS`. Once you’ve done that, you don’t need to run the slow BASIC version any more, which is a blessing, because the MS-Kermit `.BOO` file takes up to half an hour to decode using the BASIC version (depending on the system), but only seconds using `MSBPCT.EXE`.

`MSBPCT.*` There are `.BOO`-file decoders written in other languages too, like assembler, Turbo Pascal, Fortran, etc. Take your pick. They all do the same thing.

`MSBMKB.*` This is the program for encoding an `.EXE` file into a `.BOO` file. It is written in C, compiled, and translated (by itself) into `.BOO` format, suitable for decoding back into `.EXE` form by any of the MSBPCT programs. Also in other languages, including Fortran and Turbo Pascal.

`MSBHEX.*` are C programs for producing and decoding straight hex files.

1.15. Program Organization

Kermit-MS version 2 is composed of separate assembler source files, assembled separately, and linked together. The modules are:

System/Device Independent:

`MSSKER.ASM`	Main program
`MSSSEN.ASM`	File sender
`MSSRCV.ASM`	File receiver
`MSSSER.ASM`	Server operation
`MSSFIL.ASM`	File i/o
`MSSCMD.ASM`	Command parser
`MSSTER.ASM`	CONNECT command
`MSSCOM.ASM`	Packet reader and sender
`MSSSET.ASM`	SET, SHOW, and STATUS commands
`MSSSCP.ASM`	Script CLEAR, ECHO, INPUT, OUTPUT, PAUSE, TRANSMIT commands
`MSSFIN.ASM`	Dummy module for the end of the data segment; must be linked LAST.
`MSSDEF.H`	Data structure definitions and equates

System/Device Dependent:

MSG*xxx*.ASM	System-dependent graphics terminal for system *xxx*
MSU*xxx*.ASM	System-dependent keyboard translator for system *xxx*
MSX*xxx*.ASM	System-dependent code for system *xxx*
MSY*xxx*.ASM	Terminal emulation for system *xxx*
MSZ*xxx*.ASM	More terminal emulation for system *xxx*

The *xxx* is replaced by a 3-letter code for the particular system, e.g. IBM for the IBM PC family, RB1 for the Rainbow-100, etc.

The modular organization allows easier modification of the program, quicker transfer of modified portions from system-to-system. The modules are designed to be well-defined and self-contained, such that they can be easily replaced. For instance, someone who prefers windows and mice to typing commands should be able to replace the command parsing module without having to worry about the effect on the other modules.

To assemble any of the Kermit modules, file `MSSDEF.H` must be on the default disk.

All the Kermit implementations require the modules MSSCMD, MSSCOM, MSSFIL, MSSKER, MSSRCV, MSSSCP, MSSSEN, MSSSER, MSSSET, MSSTER, MSSFIN. MSSFIN *must* be linked last.

Each particular implementation requires at least an MSX*xxx* module, usually an MSU*xxx* module, and, if it is doing terminal emulation in software, also an MSY*xxx* and possible also an MSZ*xxx* module, and for graphics terminal emulation, also an MSG*xxx* module. See the batch or make files from the source distribution for details of exactly which modules are required for a particular implementation.

Once all the required object modules exist, they may be linked together to produce a Kermit program. For example, on the IBM PC:

```
A>link

   Microsoft Object Linker V2.00
(C) Copyright 1982 by Microsoft Inc.

Object Modules [.OBJ]: msscmd+msscom+mssfil+mssker+mssrcv+mssscp+msssen+
mssser+mssset+msster+msgibm+msuibm+msxibm+msyibm+mszibm+mssfin
Run File [MSSCMD.EXE]: kermit
List File [NUL.MAP]:;

    A>
```

Warning: old versions of MASM may not be able to assemble several of the large files now present in Kermit-MS. The solution is to acquire Microsoft MASM 4.0 or later.

1.16. Bringing Kermit to New Systems

You can bring Kermit-MS to MS-DOS systems that are not explicitly supported in one of two ways -- attempt to run the "generic" MS-DOS Kermit on it, or add explicit code to support your system.

To get started with Kermit on a new system, try running "generic" MS-DOS Kermit; in many cases, it will run as is. The generic version accomplishes all its port and console i/o through DOS calls, and during terminal connection does not attempt to emulate any particular kind of terminal. In some cases, the generic version may still require some fiddling to run on a new system; for instance, different systems refer to their communication ports in different ways -- COM1, J1, AUX, etc. The SET PORT command allows you to specify the port using any of these device names, or using DOS file handles -- keep trying until you find the one that works. Generic MS-DOS Kermit will probably run no faster than 1200 baud, and it only works with DOS 2.0 or later.

If you want to write code to explicitly support a new system, first call or write Kermit Distribution at Columbia to make sure no one else is already doing the same work. If you're the first, then begin by reading the file

MSXAAA.DOC, provided with the MS-DOS Kermit sources in the Kermit distribution, which is a guide to the system dependent modules of Kermit-MS. Then create new MSU*xxx*.ASM and MSX*xxx*.ASM modules, and, if your version is also doing terminal emulation in software, also an MSY and possibly an MSZ module patterned after those that have been written for other systems.

1.17. Kermit-MS VT102 Terminal Emulator Technical Summary

1.17.1. Treatment of Inbound Characters During Terminal Emulation

The following sections summarize the Kermit-MS keyboard and screen operation during emulation of H19, VT52, and VT102 terminals, prinicipally for the IBM PC but also used by the NEC APC3, Victor 9000, and Sanyo 55x systems.

Many things can happen to a character that arrives at the communication port before you see it. The sequence of events for version 2.32 is summarized in the following picture. 2.31 is similar except for no national characters.

```
                character from serial port or network
                                |
                                v
                        NUL: discard unless DEBUG is ON
                        DEL: discard unless DEBUG is ON or doing Tektronix
                                |
                        character available to rest of Kermit
                                |
                                v
        TERMINAL NONE   <-------*-------*---> to packet reader
        DISPLAY 8-BIT?          |       |
        Yes     No              v       v
        |       |               |       |
        |       remove 8-th bit |       Script routines.
        |       |               |       remove parity bit, if parity used.
        ---------               |       apply SET TRANSLATION INPUT table,
        |                       |        if active.
        apply SET TRANSLATION   |       |
        INPUT, if active        |       LOG SESSION active?
        |                       |       No      Yes
        copy char to printer?   |       |       |
        No      Yes             |       |       DEBUG ON or DISPLAY 8-BIT?
        |       |               |       |       Yes     No
        |       print char      |       |       |       |
        |       via DOS         |       |       |       remove 8-th bit
        |       |               |       |       |       |
        ---------               |       |       ---------
        |                       |       |           |
        log char if             |       |           |
        LOG SESSION active.     |       |           |--> log character
        |                       |       v           v
        v                       |       -------------
    DEBUG ON?                   |               |
    No    Yes                   |       DISPLAY 8-BIT?
    |     |                     |       Yes     No
    |   show chars, with                |       |       |
    |   tilde/caret modifiers   |       |       remove 8-th bit
    |     |                     |       |       |
    |   Exit                    |       v       v
    |                           |       -------------> character to scripts
    display char via DOS        |
    Exit                        |
                                v
                        VT102 terminal emulator
                                |
                           SET DEBUG ON?
                                No      Yes --> display characters:
                                |               show leading tilde if bit 8
                                |               set; for lower 7 bits show
                                |               caret-char if Control code.
                                |               |
                                |               Exit.
                                v
                Printing Transparently? (ESC [ 4/5 i)
```

```
                No              Yes
                |               |
                |               log character if LOG SESSION is active,
                |               send char to printer but not to screen.
                |               |
                |               Exit.
                v
        SET DISPLAY 8-BIT?
        Yes             No
        |               |
        |       remove 8-th bit of character
        |               |
        v               v
        -----------------
                |
                v
        Control character?
        Yes             No
        |               |
        |               Doing escape sequence?
        |               No              Yes
        |               |               |
        v               v               |
        -----------------               |
                |                       |
                v                       |
        do SET TRANSLATION INPUT,       |
        if active.                      |
        |                               v
        ---------------------------------
                        |
                        v
                 NUL or DEL character?
                        No      Yes --> Exit (discard)
                        |
                        |
                log character if LOG SESSION is active.
                        |
                Control character?
                Yes             No
                |               |
          do control ops        map character according to
                |               active character-set pointer. (See below)
                Exit            |
                                display character on screen
                                (with double width/height, if req'd).
                                |
                                print character via DOS if
                                Controller Print is active (ESC [ ? 4/5 i)
                                or if Kermit copy screen to printer active.
                                |
                                v
                                Exit
```

Updating of the cursor position is automatic and can be influenced by the Kermit command SET TERMINAL DIRECTION {RIGHT-TO-LEFT | LEFT-TO-RIGHT}. As a convenience, the keyboard left and right arrow keys are interchanged when the writing direction is reversed; thus, the right arrow always requests the host to move the cursor to the visual right.

The active character-set pointer is determined by two conditions:

1. The VT102 maintains two character set pointers (selectors), G0 and G1. G0 is the default pointer. Reception of Control-O selects the G0 pointer, Control-N selects the G1 pointer.

2. Which character set: US-ASCII, UK-ASCII, ALTERNATE-ROM, or line-drawing, has been assigned

to G0 and G1 pointers. The command SET CHARACTER-SET {US-ASCII, UK-ASCII, ALTERNATE-ROM} assigns that set to the G0 AND G1 pointers. Similarly, the host can assign any of the four sets to either G0 OR G1 via the escape sequences `ESC ( ` *char* or `ESC ) ` *char*, respectively, as summarized below:

`ESC ( A`	G0 points to UK symbols (ASCII with Pound Sterling sign)
`ESC ) A`	G1 points to UK symbols
`ESC ( B`	G0 points to ASCII symbols (ASCII with US pound sign #)
`ESC ) B`	G1 points to ASCII symbols
`ESC ( 0`	G0 points to special (line drawing) graphics
`ESC ) 0`	G1 points to special (line drawing) graphics
`ESC ( 1`	G0 points to ALTERNATE-ROM national characters
`ESC ) 1`	G1 points to ALTERNATE-ROM national characters
`ESC ( 2`	G0 points to special (line drawing) graphics
`ESC ) 2`	G1 points to special (line drawing) graphics

All character sets produce

```
!"  pound-sign  $%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
```

The lower case field, `'ab..yz{|}~` changes to line drawing or national characters, depending on the character set. National characters replace the character codes 60h to 7Ah (accent grave, lower case a-z) with codes 80h to 9Ah; lower case a becomes umlated u, etc in standard IBM display adapters.

DOS Code Page support

Code Pages are sets of translation tables maintained within DOS to support national languages. They affect the characters reported by the keyboard and those displayed on the screen and printer. Code Page support is loaded as device driver information in CONFIG.SYS and activated by DOS programs NLSFUNC, CHCP, KEYB, and MODE, at least under PC DOS 3.30. An EGA adapter is required for screen support; similarly, only IBM printers are discussed. Making Code Pages operate is not exactly easy, and there have been rumors that MS-DOS Code Pages from various vendors are not bug free. However, the goal is an ability to change translations for screen, keyboard, and printer by DOS commands.

Since Code Pages are the province of DOS it is clear that operations at the Bios or hardware levels will not experience Code Pages. Kermit uniformly uses DOS for printer output. Kermit CONNECT mode keyboard reading is normally done via the Bios, unless SET KEY OFF has been stated. Kermit CONNECT mode Screen reading and writing is done via both the Bios and the hardware for the VT102/VT52/Tek4010 emulators, but the terminal type of NONE uses only DOS. Thus, full Code Page support is available in Kermit by stating SET KEY OFF (use DOS) and SET TERMINAL NONE (use DOS). Outside of CONNECT mode all Kermit screen and keyboard input and output is done through DOS.

1.17.2. Keyboard Layout and Characters Sent

Here is how the keypad functions are assigned to the IBM keyboard function keys. You may change them by using the SET KEY command to define a desired key as the appropriate Kermit action verb; use SET KEY without a definition to undefine a key. Names of appropriate verbs are also shown for use in the Set Key command, such as

```
Set Key \2352 \Kbreak
```
(IBM Alt-B assigned to verb BREAK)

Verb names are system dependent, use ? in the Set Key definition part for a list of local verbs. IBM PC verbs are listed in Table 1-6; IBM key values are either straight ASCII or the IBM Bios scan code, plus 256, plus 512 for Shift key held down, plus 1024 for Control key held down, plus 2048 for Alt key held down; non-ASCII keys are always 256 decimal or greater. Keys particular to the Enhanced Keyboard have 4096 added to the result.

```
    Heath-19 and VT52 Keypads                VT102 keypad
       IBM Keys                                 IBM keys
+------+------+-------+----------+   +------+------+------+------+
| Blue |  Red |  Grey | up arrow |   |  PF1 |  PF2 |  PF3 |  PF4 |
|  F1  |  F2  |  F3   | up arrow |   |  F1  |  F2  |  F3  |  F4  |
+------+------+-------+----------+   +------+------+------+------+
|  7   |  8   |  9    |down arrow|   |  7   |  8   |  9   |  -   |
|  F5  |  F6  |  F7   |down arrow|   |  F5  |  F6  |  F7  |  F8  |
+------+------+-------+----------+   +------+------+------+------+
|  4   |  5   |  6    | rgt arrow|   |  4   |  5   |  6   |  ,   |
|  F9  |  F10 |  SF1  | rgt arrow|   |  F9  |  F10 |  SF1 |  SF2 |
+------+------+-------+----------+   +------+------+------+------+
|  1   |  2   |  3    |left arrow|   |  1   |  2   |  3   |  E   |
|  SF3 |  SF4 |  SF5  |left arrow|   |  SF3 |  SF4 |  SF5 |  n  S|
+------+------+-------+----------+   +------+------+------+  t  F|
|  0------0   |  .    |  Enter   |   |  0------0   |  .   |  e  6|
|  SF7        |  SF8  |  SF6     |   |   SF7       |  SF8 |  r   |
+-------------+-------+----------+   +-------------+------+------+
```

SF1 means push Shift and F1 keys simultaneously

CURSOR KEYS:

```
                                        H-19 & VT52          VT102
VT52/H19 key   IBM Verb   IBM key        All Modes       Numeric   Application

up arrow       UPARR     up arrow        ESC A           ESC [ A   ESC O A
down arrow     DNARR     down arrow      ESC B           ESC [ B   ESC O B
right arrow    RTARR     right arrow     ESC C           ESC [ C   ESC O C
left arrow     LFARR     left arrow      ESC D           ESC [ D   ESC O D
```

AUXILIARY KEYPAD:

```
                                       Heath-19 & VT52          VT102
VT52/H19 key   IBM Verb   IBM key      Numeric Applic.     Numeric    Applic.

PF1/HF7/Blue   GOLD,PF1    F1          ESC P   ESC P       ESC O P    ESC O P
PF2/HF8/Red    PF2         F2          ESC Q   ESC Q       ESC O Q    ESC O Q
PF3/HF9/Grey   PF3         F3          ESC R   ESC R       ESC O R    ESC O R
PF4/HF1        PF4         F4          ESC S   ESC S       ESC O S    ESC O S
0              KP0         SF7         0       ESC ? p     0          ESC O p
1              KP1         SF3         1       ESC ? q     1          ESC O q
2              KP2         SF4         2       ESC ? r     2          ESC O r
3              KP3         SF5         3       ESC ? s     3          ESC O s
4              KP4         F9          4       ESC ? t     4          ESC O t
5              KP5         F10         5       ESC ? u     5          ESC O u
6              KP6         SF1         6       ESC ? v     6          ESC O v
7              KP7         F5          7       ESC ? w     7          ESC O w
8              KP8         F6          8       ESC ? x     8          ESC O x
```

```
9               KP9       F7        9       ESC ? y  9        ESC O y
comma  (,)      KPCOMA    SF2       ,       ESC ? l  ,        ESC O l
minus  (-)      KPMINUS   F8        -       ESC ? m  -        ESC O m
period (.)      KPDOT     SF8       .       ESC ? n  .        ESC O n
Enter           KPENTER   SF6       ^M(cr)  ESC ? M  ^M       ESC O M
```

(SFn means hold down Shift key while pressing Function key n.)

An often confusing item is knowing the mode of the auxillary keypad: numeric or application. Digital Equipment Corporation designed the terminal to change modes only under command from the remote computer and not at all from the keyboard. So the startup state is numeric/cursor mode, and reception of escape sequences ``ESC [ ? 1 h'' or ``l'' changes the mode. Kermit verbs for the keypad and cursor keys generate the correct escape sequences appropriate to the current mode and terminal type.

A best attempt is made to safely test for the 101/102 key Enhanced keyboard and use it if present. If it is present then the keyboard translator separates the individual arrow keys from those on the numeric keypad and also separates the asterisk and forward slash keys on the keypad from those on the regular typewriter keyboard. These special Enhanced keyboard keys are reported as scan codes with 4096 added to the base scan code.

OTHER IBM KEYS OPERATIONAL IN CONNECT MODE:

```
IBM key          IBM Verb      Action

Keypad Del                     Send ASCII Del code (rubout) \127
Backspace (<-)                 Send ASCII Del code (rubout) \127  (BS is \8)
Keypad -         MODELINE      Toggle mode line on/off (only if Mode Line is
                               enabled and not used by the host).
Alt -            TERMTYPE      Toggle among H-19, VT52, and VT100 emulations.
Alt =            RESET         Clear screen and reset terminal emulator to
                               starting (setup) state.
Alt B            BREAK         Send a BREAK signal
Alt H            HELP          Show drop down help menu (detailed below)
Alt S            STATUS        Show settings
Alt X            EXIT          Exit Connect mode, back to Kermit prompt

Home             HOMSCN        Roll screen up (text down) to beginning of
                                storage.
End              ENDSCN        Roll screen down (text up) to end of storage.
PgUp             UPSCN         Roll screen up (back, earlier) one screen.
PgDn             DNSCN         Roll screen down (forward, later) one screen.
Ctrl-PgUp        UPONE         Roll screen up one line.
Ctrl-PdDn        DNONE         Roll screen down one line.

Control PrtSc    PRTSCN        Toggle  on/off copying of received text to
                               printer, "PRN" shows on far right of mode
                               line when activated.

Control-End      DUMP          Dump image of screen to a disk file or device.
                               Default filename is KERMIT.SCN in the current
                               directory. Use command SET DUMP to change the
                               filename.  Screen images are appended to the
                               file, separated by formfeeds.

Shift-PrtSc      Standard DOS Print-screen, dump screen image to printer.

unassigned       HOLDSCRN      DEC style Holdscreen, same as typing Control-S.
```

"Alt -" means hold down Alt and type minus on the upper key rank. This switches among the various kinds of emulation but does not change most operating parameters of the emulator.

CONNECT ESCAPE COMMANDS:

Type the Kermit escape character (normally ‘‘^]’’), then one of the keys below:

```
                                                         (equivalent IBM Verb)
     ?   display this short list.                                HELP
     0   send a null character.                                  NULL
     B   send a BREAK signal.                                    BREAK
     C   close connect session & return to Kermit prompt.        EXIT
     F   dump screen to filespec, default is KERMIT.SCN.         DUMP
     H   hangup the phone or network connection                  HANGUP
     L   send a Long BREAK signal                                LBREAK
     M   toggle mode line on/off.                                MODELINE
     P   push to DOS.                                            DOS
     Q   quit (suspend) logging.                                 LOGOFF
     R   resume logging.                                         LOGON
     S   show status.                                            STATUS
     Kermit escape character itself: send it to the host.
```

1.17.3. Responses To Characters Received By the Terminal Emulator

Spaces shown between characters of escape sequences are there for ease of reading. The actual sequences contain no spaces. Unknown escape sequences of the form "ESC char" are absorbed by the emulator without further effect; longer unknown escape sequences echo the extra characters.

DEC VT102 functions while in ANSI (VT102) mode, unsupported features marked by an asterisk (*):

```
 Escape Seq      Mnemonic       Description of Action

 ESC D           IND            Index, moves cursor down one line, can scroll
 ESC E           NEL            Move cursor to start of line below, can scroll
 ESC H           HTS            Set one horizontal tab at current position
 ESC M           RI             Reverse Index, cursor up one line, can scroll
 ESC Z           DECID          Identify terminal (response is ESC [ ? 6 c)
 ESC c           RIS            Reset terminal to initial state
 ESC =           DECKPAM        Enter keypad application mode
 ESC >           DECKNPNM       Enter keypad numeric mode
 ESC 7           DECSC          Save cursor position and attributes
 ESC 8           DECRC          Restore cursor from previously saved position
 ESC # 3         DECDHL         Double height and width line, top half
 ESC # 4         DECDHL         Double height and width line, bottom half
 ESC # 5         DECSWL         Single height and width line
 ESC # 6         DECDWL         Double width single height line
 ESC # 8         DECALN         Test screen alignment, fill screen with E’s
 ESC [ Pn @      ICH            ANSI insert Pn spaces at and after cursor
 ESC [ Pn A      CUU            Cursor up Pn lines, does not scroll
 ESC [ Pn B      CUD            Cursor down Pn lines, does not scroll
 ESC [ Pn C      CUF            Cursor forward, stays on same line
 ESC [ Pn D      CUB            Cursor backward, stays on same line
 ESC [ Pn; Pn H CUP             Set cursor to row, column (same as HVP)
 ESC [ Ps J      ED             Erase in display:
                                 0 = cursor to end of screen, inclusive
                                 1 = start of screen to cursor, inclusive
                                 2 = entire screen, reset lines to single
                                     width, cursor does not move.
 ESC [ Ps K      EL             Erase in line:
                                 0 = cursor to end of line, inclusive
                                 1 = start of line to cursor, inclusive
                                 2 = entire line, cursor does not move
 ESC [ Pn L      IL             Insert Pn lines preceding current line.
 ESC [ Pn M      DL             Delete Pn lines from current downward, incl.
 ESC [ Pn P      DCH            Delete Pn chars from cursor to left, incl.
 ESC [ Pn; Pn R CPR             Cursor report (row, column), sent by terminal
                                Example: home position yields ESC [ 1; 1 R
```

```
ESC [ Pn c      DA              Device attributes (reports ESC [ ? 6 c)
ESC [ Pn; Pn f HVP              Set cursor to row, column (same as CUP)
ESC [ Ps g      TBC             Tabs clear, 0 = at this position, 3 = all
ESC [ 4 h       IRM             Insert mode on
ESC [ 20 h      LNM             Set newline mode (cr => cr/lf)
ESC [ 4 l       IRM             Replacement mode on
ESC [ 20 l      LNM             Reset newline mode (cr => cr)
ESC [ ? Ps;...;Ps h  SM         Set mode, see table below
ESC [ ? Ps;...;Ps l  RM         Reset mode, see table below
        Ps      Mnemonic        Mode            Set (h)         Reset (l)
        0                       error (ignored)
        1       DECCKM          cursor keys     application     cursor/numeric
        2       DECANM          ANSI/VT52       ANSI/VT102      VT52
        3       DECCOLM         Columns         +132 col        80 col
        4       DECSCLM        *Scrolling       smooth          jump
        5       DECSCNM         Screen          reverse video   normal
        6       DECOM           Origin          relative        absolute
        7       DECAWM          Autowrap        on              off
        8       DECARM         *Autorepeat      on              off
        9       DECINLM        *Interlace       on              off
        18      DECPFF          Printer termination character, use FF if set
        19      DECPEX          Printer extent,set=screen,off=scrolling region
        34      n/a             Invoke macro:  TERMINALS       TERMINALR
        38      n/a             Graphics (Tek) ++graphics      text
                              + See comments on EGA boards.
                             ++ Ignored if DISABLE TEK has been given.
ESC [ Pn i      MC              Printer controls (Media Copy)
        0                        Print whole Screen
        4                        Exit printer controller (transparent print)
        5                        Enter printer controller (transparent print)
ESC [ ? Pn i    MC              Printer controls (Media Copy)
        1                        Print line containing cursor
        4                        Exit auto print (stop echoing to printer)
        5                        Enter autoprint (echo screen chars to printer)
ESC [ Ps;...;Ps m  SGR          Select graphic rendition
                                 0 = all attributes off (#'s 1, 4, 5, 7)
                                 1 = bold, intensify foreground
                                 4 = underscore (reverse video on IBM CGA)
                                 5 = blink
                                 7 = reverse video
                non-DEC extensions: 30-37 = foreground color = 30 + colors
                                    40-47 = background color = 40 + colors
                                       colors: 1 = red, 2 = green, 4 = blue
ESC [ Ps n      DSR             Device Status Report.
                                Response from VT100: 0=ready, 3=malfunction.
                                Command to VT100: 5=report status with DSR,
                                6=report cursor position using CPR sequence.
ESC [ Ps;...;Ps q  DECLL        Load LEDs, Ps = 0 means clear LED #1-4
                                Ps = 1,2,3,4 sets LED # 1,2,3,4 on status line.
ESC [ Pn; Pn r DECSTBM          Set top and bottom scrolling margins, resp.
                                ESC [ r resets margin to full screen.
ESC [ sol x     DECREQTPARM     Request terminal parameters, see table below
ESC [ sol; par; nbits; xspeed; rspeed; clkmul; flags x
                DECREPTPARM     Reports terminal parameters
                                sol = 0  request; terminal can send unsolicited
                                  reports - supported as sol = 1 below.
                                sol = 1, request; term reports only on request
                                sol = 2, this is a report (DECREPTPARM)
                                sol = 3, terminal reporting only on request
                                par = 1 none, 2 space, 3 mark, 4 odd, 5 even
                                nbits = 1 (8 bits/char), 2 (7 bits/char)
                                xspeed,rspeed = transmit & receive speed index
  0,8,16,24,32,40,48,56,64,72,80,88,96,104,112,120,128 correspond to speeds of
```

```
 50,75,110,134.5,150,200,300,600,1200,1800,2000,2400,3600,4800,9600,19200,
 and 38400 baud.              clkmul = 1 (clock rate multiplier is 16)
                              flags = 0-15 (Setup Block #5), always 0 here
ESC [ 2; Ps y  DECST          *Confidence tests - not supported
               SCS            Select character sets.
ESC ( A        SCS            G0 points to UK symbols
ESC ) A        SCS            G1 points to UK symbols
ESC ( B        SCS            G0 points to ASCII symbols
ESC ) B        SCS            G1 points to ASCII symbols
ESC ( 0        SCS            G0 points to special (line drawing) graphics
ESC ) 0        SCS            G1 points to special (line drawing) graphics
ESC ( 1        SCS            G0 points to alt char ROM - national symbols
ESC ) 1        SCS            G1 points to alt char ROM - national symbols
ESC ( 2        SCS            G0 points to alt graphics ROM - as ESC ( 0
ESC ) 2        SCS            G1 points to alt graphics ROM - as ESC ) 0
                               (Separate graphics used for DEC and Heath)
^E             ENQ           *Answerback message (not supported)
^G             BELL           Sound VT102 style beep
^H             BS             Backspace, move cursor left one character
^I             HT             Horizontal tab, move cursor to next tabstop
^J             LF             Linefeed, move cursor down one line
^K             VT             Vertical Tab, treated as a line feed
^L             FF             Formfeed, treated as a line feed
^M             CR             Carriage return, move cursor to col 1
^N             SO             Select usage of G1 character set
^O             SI             Select usage of G0 character set
^X             CAN            Cancel escape sequence in progress
^Z             SUB            Treated as a CAN

Other extensions:
ESC [ 25; Pc f                VT52/VT100 move cursor to 25th line.
ESC [ 25; Pc H                VT52/VT100 move cursor to 25th line.
                              (These will disable Kermit’s own status line.)
ESC * char                    VT200 series graphics command, ignored.
ESC ^L                        Enter Tektronix sub-mode, clear Tek screen.
                              (This is ignored if DISABLE TEK has been given)
```

1.17.4. DEC VT102 Functions While in VT52 Mode

```
Escape sequence               Description of action

ESC A                         Cursor up
ESC B                         Cursor down
ESC C                         Cursor right
ESC D                         Cursor left
ESC F                         Enter graphics mode
ESC G                         Exit graphics mode
ESC H                         Cursor home
ESC I                         Reverse line feed
ESC J                         Erase to end of screen
ESC K                         Erase to end of line
ESC V                         Print cursor line
ESC X                         Exit Printer Controller mode, transparent print
ESC Y row column              Direct cursor address, offset from space
ESC W                         Enter Printer Controller mode,transparent print
ESC Z                         Identify (response is ESC / Z)
ESC ^ (caret)                 Enter autoprint mode (printer echoes screen)
ESC _ (underscore)            Exit autoprint mode
ESC ]                         Print Screen
ESC =                         Enter alternate keypad mode
ESC >                         Exit alternate keypad mode
ESC <                         Enter ANSI mode (changes to VT102)
```

1.17.5. Heath-19 Functions While in Non-ANSI Mode

Escape seq	Mnemonic	Description of action
ESC A	HCUU	Cursor Up
ESC B	HCUD	Cursor Down
ESC C	HCUF	Cursor Forward, stays on same line
ESC D	HCUB	Cursor Backward, stays on same line
ESC E	HCD	Clear display
ESC F	HEGM	Enter Graphics mode
ESC G	HXGM	Exit Graphic mode
ESC H	HCUH	Cursor Home
ESC I	HRI	Reverse Index
ESC J	HEOP	Erase to end of page
ESC K	HEOL	Erase to end of line
ESC L	HIL	Insert line
ESC M	HDL	Delete line
ESC N	HDCH	Delete character
ESC O	HERM	Exit Insert Char mode
ESC Y row col	HDCA	Direct cursor addressing, offset from space
ESC Z	HID	Identify (response is ESC / K which is a VT52)
ESC b	HBD	Erase Beginning of display
ESC j	HSCP	Save cursor position
ESC k	HRCP	Set cursor to saved position
ESC l	HEL	Erase entire line
ESC n	HCPR	Cursor Position Report request
ESC o	HEBL	Erase beginning of line
ESC p	HERV	Enter Reverse Video mode
ESC q	HXRV	Exit Reverse Video mode
ESC r Bn	HMBR	*Modify baud rate - not supported
ESC t	HEKS	*Enter Keypad shifted mode, not supported
ESC u	HXKS	*Exit Keypad shifted mode, not supported
ESC v	HEWA	Wrap around at end of line
ESC w	HXWA	Discard at end of line
ESC x Ps	HSM	Set Mode. See table below
ESC y Ps	HRM	Reset Mode. See table below

Ps	Mnemonic	Mode	Set (x)	Reset (y)
1	HSM/HRM	25th line	enabled	+disabled
2		*keyclick	off	on
3		*holdscreen	enabled	disabled
4		cursor type	block	underline
5		cursor on/off	on	off
6		*keypad-shifted	shifted	unshifted
7		alt app keypad	enabled	disabled
8		*linefeed	lf=>cr/lf	lf=>lf
9		newline mode	cr=>cr/lf	cr=>cr

+ disabling the 25th line also clears it

Escape seq	Mnemonic	Description of action
ESC z	HRAM	Reset to power-up configuration
ESC =	HAKM	Enter Alternate Keypad mode
ESC >	HXAM	Exit Alternate Keypad mode
ESC <	HEAM	Enter ANSI mode (ESC [stuff)
ESC @	HEIM	Enter Insert Char mode
ESC [	HEHS	*Enter Hold Screen mode, not supported
ESC \	HXHS	*Exit Hold Screen mode, not supported
ESC { and }	HEK, HDK	*Keyboard enable/disable, not supported
ESC]	HX25	*Transmit 25th line, not supported
ESC #	HXMP	*Transmit page, not supported

1.17.6. Heath-19 Functions While in ANSI Mode

Escape Seq	Mnenonic	Description of Action
ESC [s	PSCP	Save cursor position & attributes
ESC [u	PRCP	Restore cursor position & attributes
ESC [z	PRAM	Reset to power-up configuration
ESC [2 J	ED	Erase entire screen but do not move cursor; regular Heath-19 moves cursor to Home.
ESC [? 2 h	PEHM	Revert to normal Heath-19 non-ANSI mode
ESC [> Ps h	SM	Same as ESC x Ps
ESC [> Ps l	RM	Same as ESC y Ps

Plus most of the ANSI escape sequences listed for the VT102.

1.17.7. Tektronix 4010/4014 Graphics Terminal Functions

MS-Kermit's Tektronix 4010 emulator responds to ordinary text, several special control codes (for drawing lines and dots), and several escape sequences, as shown in Table 1-10. The commands SET DEBUG and SET TRANSLATION INPUT are effective in Tek mode.

Control Code		Action
FS, Control-\	Backslash	draw dots
GS, Control-]	Right square bracket	draw lines
RS, Control-^	Caret	Draw dots incrementally
US, Control-_	Underscore	Display text
BEL, Control-G		Beep, make a noise
BS, Control-H		Backspace, non-destructive
HT, Control-I		Tab, convert to single space
LF, Control-J		Line feed, go down one line
VT, Control-K		Move up one text line
FF, Control-L		Clears the screen
CR, Control-M		Carriage return, start of line
CAN, Control-X		Exit Tek sub-mode, or ignore
DEL, RUBOUT		Delete code, same as BS

Escape Sequence	Action
ESC Control-E	Send a status report, turn on Bypass mode
ESC Control-L	Clear the screen (enter sub-mode from VT102)
ESC Control-X	Turn on Bypass mode
ESC Control-Z	Activate crosshairs (GIN mode) and Bypass mode
ESC Z	Send terminal identification
ESC ' (accent grave)	Use solid lines in drawing
ESC a through ESC e	Use dashed line patterns: a=fine dots, b=short dashes c=dash dot, d=long dash dot e=dash dot dot.
ESC [Pn ; Pn m	Set ANSI colors. Same as for VT102.
ESC [? 3 8 l	Exit Tek mode (become text terminal, VT102 etc)
ESC [? 3 8 h	Enter Tek mode (from VT102 mode)

Table 1-10: Response of MS-Kermit Tektronix Emulator to Received Characters

In the table, US is the name for the ASCII character Control-Underscore, 31 decimal. Text is written starting with the last drawn point being the lower left corner of the first 8 by 8 character cell. The drawing position is updated by 8 dots to the right for each character, and lines wrap at column 80 (column 90 for Hercules boards). If text extends "below the screen" the sign "More >" is shown at the bottom right corner and the user needs to press a key to continue. Then the screen will be cleared and the new text will start at the top of the screen (no scrolling is done in graphics mode). A real Tek 4010 begins new text at column 40 and will overwrite dots from older material. The high resolution EGA screen and the Hercules screen will hold 43 lines, the CGA and Monochome screens hold 25 lines, and the AT&T screen holds 50 lines. Hercules screens are 90 characters wide and others are 80 characters wide. Monochrome systems lack graphics so the text is the normal hardware character font placed at the nearest normal 80x25 location (similarly, "drawing" on Monochrome systems is achieved by using a text plus ("+") sign where a dot would appear). Text mode is interrupted by the drawing commands discussed below.

Bypass Mode:

Certain Tektronix commands turn on or off "Bypass" mode whereby incoming text is not displayed on the screen. Removal of echos of the GIN mode, discussed below, is the major use of Bypass. Bypass mode is turned on by receipt of ESC Control-E, ESC Control-X, and ESC Control-Z and it is turned off upon receipt of BEL, LF, CR, US, other escape sequences, and resetting the terminal.

Drawing commands GS, FS, RS:

1. Draw a line or move to a point: GS <*xy xy . . . xy*>

GS is the name for ASCII character Control-] (right square bracket), decimal 29. <*xy*> stands for an encoded x,y coordinate as explained below. One or more x,y coordinates may follow GS and line segments are drawn from point to point. The first point is reached without drawing so that GS and the initial <*xy*> is a simple "move-to" command rather than a "draw-to" command. Lines may be constructed from six dash patterns described in Table 1-10. <*xy*> coordinates are encoded by separating the 10 bit value of x and of y into 5 bit components and then adding two high bits to each to identify which component is being represented: high-y, low-y, high-x, or low-x. They are transmitted in that order, with the low-x byte always sent last. In fact, bytes may be omitted if they do not change from point to point, provided that low-x is always sent. These bytes range from ASCII space (32 decimal) to ASCII DEL (127 decimal). Details are given below, and summarized in Table 1-12. This mode completes when a new command or a CR LF (carriage return, line feed) arrives; escape sequences are processed transparently but other control codes are ignored. The interrupting character is accepted and processed next.

2. Draw dots at given locations: FS <*xy xy . . . xy*>

FS is the name for the ASCII character Control-\ (backslash), decimal 28. <*xy*> is in the same form as above. A dot is drawn at each x,y point. This mode completes when a new command or a CRLF character arrives; escape sequences are processed transparently but other control codes are ignored. The interrupting character is accepted and processed next.

3. Draw dots from the current location: RS <*pen*> <*direction*> <*direction*> . . . <*direction*>

RS is the name for the ASCII character Control-^ (caret), decimal 30. *pen* is the character Space (32 decimal) to move without drawing or P (80 decimal) to draw while moving. <*direction*> is one of the letters A, E, D, F, B, J, H, I as shown in Table 1-11.

Example: RS P J J J (no spaces here, naturally) means draw three dots in the southwest direction, stepping to each in turn. This mode completes when a new command or a non-<*pen*> or non-<*direction*> character arrives; the interrupting character is accepted and processed next.

direction	*Move One Tek Dot This Way*				
A	East (right)				
E	East and North	F	D	E	
D	North (up)				
F	North and West	B	*	A	(* is current location)
B	West				
J	South and West	J	H	I	
H	South				
I	South and East				

Table 1-11: Tektronix Dot-Drawing Commands

Graphics INput (GIN) mode:

Graphics input mode is entered when ESC Control-Z is received. A crosshair is drawn on the screen and may be moved by the numeric keypad arrows (fine scale motion) or the Shift key and these arrows (coarse scale motion). Pressing an ASCII-producing key sends the position of the crosshairs to the host as the sequence of: pressed key, X coordinate, Y coordinate, carriage return, then removes the crosshairs, and then returns to text mode. The coordinates are encoded by splitting them into five bit fields, adding an ascii space (20H) to each, and are sent as high-y, low-y, high-x and low-x bytes. Bypass mode is active while the report is sent to supress echos of the report. One may prematurely exit GIN mode by typing Control-C or Control-Break. Shift-PrtSc (DOS screen dump) remains active, however.

Status or Position Report:

ESCAPE Control-E requests a status report from the emulator. Tek terminals have many sub-fields. Kermit-MS sends a byte of 24 hex for being in text mode or 20 hex otherwise, followed by the encoded X then Y coordinates and a carriage return. Coordinates are encoded 5 bits at a time similar to the GIN report.

Identification Report:

ESCAPE Z requests terminal identification, as for VT52 and VT102. Currently this report is the 10 character sequence `IBM_TEK ESCAPE / Z` (no spaces).

Screen Capturing:

Kermit does not implement a graphics screen capture facility. There are many such Terminate-and-Stay-Resident (TSR) programs in circulation, as either public domain offerings or parts of commercial packages (Paint programs and even `GRAPHICS.COM` from DOS). High resolution EGA screens require more than the `GRAPHICS.COM` program. MS Windows tells the program (Kermit-MS) the system is using a pure text-only monochrome adapter so dots are shown as plus signs.

Although Kermit cannot save graphics screens directly (e.g. via the `^]F` connect-mode command), the received Tektronix escape sequences can still be logged to a PC file using the LOG SESSION command. The resulting log cannot be "played back" directly on the PC, but it can be transferred to the host and run through Kermit's Tek emulator again, just like a character-mode Kermit session log.

VGA Modes:

Considerable effort went into ensuring the graphics display would work automatically and not damage monitors. Thus, Kermit-MS safely tests the active display adapter for its kind and capabilities before starting graphics mode. Recent VGA and EGA+ display boards are capable of the 640 by 480 scan-line 16-color "VGA" mode which is now available on IBM PS/2 computers. The Tek emulator will happily run with 480 scan lines, but: the normal 256KB of video memory is sufficient to save only the top 407 lines of the graphics image. So activating this higher resolution mode is accomplished by the command SET TERMINAL GRAPHICS VGA and is not done automatically (the VGA is used in EGA mode). The 320 by 200 line by 256 color MCGA mode has too coarse a resolution for graphics line drawing and is not supported by Kermit.

Coordinate Encoding:

Coordinate 0,0 is the lower left corner and the X axis is horizontal. Tektronix positions are mapped into the typically 640 dots wide by 200 or 350 dots high PC screen and thus adjacent Tek positions may yield the same PC screen dot.

4010-like devices use positions from 0 to 1023 for both X and Y, although only 0 to 779 are visible for Y due to screen geometry. The Tek screen is 10.24 by 7.80 inches and coordinates are sent as 1-4 characters.

4014-like devices use positions 0 to 4095, but each movement is a multiple of 4 positions unless the high-resolution LSBXY are sent. This makes it compatible with the 4010 in that a full sized plot fills the screen. The emulator accepts the LSBXY components but does not use them.

The various modes are summarized in Table 1-12, in which the following notation is used:

HIX, HIY = High order 5 bits of a 10 or 12 bit position.
LOX, LOY = Middle order 5 bits of position (low order of Tek 4010).
LSBXY = Low order 2 bits of X + low order 2 bits of Y (4014 mode), recognized by the Tek emulator but not used to calculate position.

Hi Y	Lo Y	Hi X	LSBXY	Characters Sent (Lo-X Always Sent)					
Same	Same	Same	Same					Lo-X	
Same	Same	Same	Diff		LSB,	Lo-Y,		Lo-X	4014
Same	Same	Diff	Same			Lo-Y,	Hi-X,	Lo-X	
Same	Same	Diff	Diff		LSB,	Lo-Y,	Hi-X,	Lo-X	4014
Same	Diff	Same	Same			Lo-Y,		Lo-X	
Same	Diff	Same	Diff		LSB,	Lo-Y,		Lo-X	4014
Same	Diff	Diff	Same			Lo-Y,	Hi-X,	Lo-X	
Same	Diff	Diff	Diff		LSB,	Lo-Y,	Hi-X,	Lo-X	4014
Diff	Same	Same	Same	Hi-Y,				Lo-X	
Diff	Same	Same	Diff	Hi-Y,	LSB,	Lo-Y,		Lo-X	4014
Diff	Same	Diff	Same	Hi-Y,		Lo-Y,	Hi-X,	Lo-X	
Diff	Same	Diff	Diff	Hi-Y,	LSB,	Lo-Y,	Hi-X,	Lo-X	4014
Diff	Diff	Same	Same	Hi-Y,		Lo-Y,		Lo-X	
Diff	Diff	Same	Diff	Hi-Y,	LSB,	Lo-Y,		Lo-X	4014
Diff	Diff	Diff	Same	Hi-y,		Lo-Y,	Hi-X,	Lo-X	
Diff	Diff	Diff	Diff	Hi-y,	LSB,	Lo-Y,	Hi-X,	Lo-X	4014
Kind code for byte:				20h	60h	60h	20h	40h	
				(transmitted left to right)					

Table 1-12: MS-Kermit Tektronix Coordinate Interpretation

Note that LO-Y must be sent if HI-X has changed so that the Tektronix knows the HI-X byte (in the range of

20h-3Fh) is HI-X and not HI-Y. LO-Y must also be sent if LSBXY has changed, so that the 4010 will ignore LSBXY and accept LO-Y. The LSBXY byte is

$$60h+(MARGIN\times 10h)+(LSBY\times 4)+LSBX$$

MARGIN is 0 here and refers to splitting the screen left and right for text rollover, which the Kermit Tek emulator does not do.

Tek 4010 Example:

Suppose <*xy*> is point y = 300, x = 500 in Tektronix coordinates. Split each 10-bit coordinate into 5-bit groups, add add the Kind code to each. Send the X part last.

```
                   HI-Y     LO-Y                         HI-X     LO-X
    Y=300d=012Ch=  01001    01100   X=500d=01F4h=  01111    10100
      +Kind code +100000 +1100000     +kind code +100000 +1000000
    Binary        101001 01101100                 101111  1000100
    ASCII            )       l                       /       D
```

So <*xy*> = (500,300) is sent or received in a GS command as ‘‘`)l/D`’’. An example in C (program fragments):

```
#define ESC  27
#define GS   29
#define US   31
FILE *fp;                     /* File descriptor for terminal */
 . . .
    fputc( GS, fp); coord( 75, 65);       /* Move to 75,65      */
    fputc( ESC, fp); fputs("[31m", fp);   /* Set foreground to red */
    fputc( US, fp); fputs("A House", fp); /* Annotate at 75,65  */
    fputc( ESC, fp); fputs("[33m", fp);   /* Set foreground to yellow */
    fputc( GS, fp);                       /* Now draw lines...  */
    coord( 50, 50); coord(300, 50);       /* Bottom side        */
    coord(300,200); coord( 50,200);       /* Right wall, top    */
    coord(175,250); coord(300,200);       /* Roof               */
    fputc( GS, fp);                       /* Start a new line   */
    coord( 50, 50); coord( 50,200);       /* Left wall at 50,50 */
    fputc( ESC, fp); fputs("[37m", fp);   /* Set foreground to white */
    . . .
coord(x, y) int x, y; {       /* Send x,y coordinates to Tek 4010 */
    fputc((y / 32) + 32, fp);             /* High y */
    fputc((y % 32) + 96, fp);             /* Low  y */
    fputc((x / 32) + 32, fp);             /* High x */
    fputc((x % 32) + 64, fp);             /* Low  x */
}
```

1.18. IBM PC Kermit Technical Summaries

Under normal circumstances, MS-Kermit takes advantage of the computer’s hardware, and often bypasses DOS (sometimes even BIOS) to achieve high performance, to exercise special machine features, or to produce an attractive screen display. Thus, it is not in all respects a "well behaved" DOS program.

MS-Kermit redirects interrupts 0BH (COM2/4) or 0CH (COM1/3), 14H (serial port), 23H (Control-Break), 24H (DOS Critical Error) and returns them when done. It uses the BIOS for keyboard, video display, and system information interrupts. It examines segment 40H for EGA operating modes and it does direct screen reads and writes. Memory for the screen roll backbuffer is negotiated with DOS to leave room for a second copy of `COMMAND.COM` to run tasks within Kermit; about 100KB to 148KB is needed for the entire program. Video page zero is normally used, but page one is employed to save screens with non-standard dimensions. Hercules and other graphics mode displays are supported only in Tektronix terminal mode. Kermit’s timing delays are dynamically adjusted each time the serial port is started to accomodate machines of different speeds; duration of the normal software timing loop is measured with the hardware timer chip and looping is adjusted to produce uniform delays on 8088 through 80386 machines.

1.18.1. Kermit-MS/IBM on Local Area Networks

The IBM version of Kermit-MS has support for the IBM Local Area Network NetBIOS (and emulators) interface, Interrupt 5CH, with additional support for selected vendor specific features (presently just AT&T STARLAN), activated by the SET PORT NET command, described above, direct support for the Ungermann Bass Net One proprietary Interrupt 14h interface, and via SET PORT BIOSn support for many other networks which intercept the Bios serial port interrupt 14h. Communications across a LAN occurring through the NetBIOS interface use virtual circuits (Sessions), named nodes, and conventional NetBIOS packets. Kermit-MS does not use LAN terminal interface packages nor the Redirector or similar functions.

Kermit LAN operations are harmonious with normal network activity and many pairs of Kermits can communicate simultaneously. Kermit does not use LAN File Server functions, since these are proprietary and vendor-specific. Kermit can, however, send and receive files to/from a LAN file server.

Since Kermit uses the standard NetBIOS interrupt 5CH interface, it will run on most LANS including IBM PC Net, IBM Token Ring, AT&T STARLAN, and many others, and will run with Novell NetWare software. Presently, Kermit knows some details of STARLAN and is able to send a BREAK across the net and can use ISN node names with long path parts. If STARLAN is not operating these features are not available. As more detailed information becomes available special features of other networks can be built-in.

The sequence of operations is similar for a client or server Kermit. The SET PORT NET command is issued by both. This command causes Kermit to validate the presence of the Interrupt 5CH interface, test for vendor additions, test for a session already underway, establish and display a unique Kermit node name, but not make a network session. The node name of the remote server machine follows the word NET; this is not to be confused with our own node name discussed below.

If an earlier LAN session is still active then the current remote node name field of the command is examined for presence of a name. If a name is given then Kermit asks the user whether to RESUME the session or start a NEW one. Starting a new one results in Kermit hanging up the old session (HANGUP) before proceeding; resuming an old one requires no further work at this point.

When Kermit attaches to the network for the first time it needs to select a unique local node name so that two systems can form a Session by using these names as addresses. Kermit uses a simple algorithm to make the name. Kermit probes the network adapter board/software for the name of the local system. If the name is present Kermit makes its own name by appending a dot K (.K) to the local name. If the local name is absent then Kermit first tries a standard name of "mskermit.K"; should the network report that the name is not unique (another node is using the name) then the user is asked to choose a name. This process continues until a unique name is obtained or the user decides to quit. The final Kermit node name is reported on the screen; client Kermits will need to know the name of the server Kermit.

Communication across the LAN begins differently for client and server Kermits. The server must be started first, by simply placing a Kermit in server mode. This results in a network Listen request being posted so that arriving packets with the correct node name can be delivered to the server Kermit. Next, a client Kermit tries to connect to the server by issuing a Kermit server command to the proper node name (as given in the client's SET PORT NET node command); REMOTE WHO is a satisfactory choice. The client machine actually issues a network Call to the server's node name to make a connection and then follows it with data packets holding the Kermit server request. The initial exchange of packets establishes a particular virtual circuit between the two nodes. If the connection cannot be started then the client Kermit reports this fact to the user. The most common causes of a failure at this point are:

1. The client Kermit did not specify the correct server Kermit node name (spelling errors, wrong case for

letters, missing dot K),

2. One or both machines are using a network adapter board which is not the first in the machine; Kermit uses only the first board,

3. The LAN NetBIOS emulator does not fully support IBM standard virtual circuits,

4. The server machine was not started on the network before the client.

A virtual circuit will be broken if a sender or receiver gets no response to a request within a short time interval set by the LAN hardware/software. However, the LAN procedures within Kermit automatically reestablish the circuit transparently to the user when new information is communicated; the last used remote node name is remembered internally for this purpose. This also means the server Kermit will respond to a connection from a new client Kermit if the first client is idle for say a minute or so. A session can be terminated by the user by issuing the HANGUP command or by exiting Kermit. A session will not be broken this way if the user on the client Kermit changes to a regular serial port.

Finally, when Kermit returns control to DOS, but not via the PUSH command, its unique Kermit node name is removed from the network adapter board.

During network communications Kermit uses network packets holding 256 bytes of data. If both Kermits are given the command

```
SET RECEIVE PACKET 1000
```

then the network and Kermit will be used to best efficiency. Experience has shown that the client Kermit should have its TIMER OFF because the server may be asked to do an operation via DOS which does not complete before the client side would timeout. An observation of some token passing networks indicates that Kermit packets slightly longer than 256, 512, etc bytes result in marked slowing down because the remaining small piece is not sent until a net timer expires. Carrier sense (Ethernet, STARLAN) boards seem to be more agressive and export small packets immediately.

Support for the Ungermann-Bass Net/One network, with its NET Command Interface (NETCI), was contributed by Renne Rehmann and Henrik Levkowetz. In addition to the SET PORT NET [nodename] command, which may be used to connect to other nodes on the net with the standard NetBIOS calls, NETCI provides the means to connect directly to serial ports on the Ungermann-Bass network. Use SET PORT UB-Net1 and enter Connect mode. The NETCI prompt, >>, should appear and all the usual NETCI commands (connect, get, list, resume, abandon, examine, identify, set, logout, quit) may be selected. This line is disconnected when Kermit exits. However, the line may be put on hold, exit Kermit, then later restart Kermit and give the SET PORT UB-Net1 and CONNECT commands, and Resume the line.

Some LANs intercept the normal serial port Bios interrupt 14H and masquerade as a modem. This service can be engaged within Kermit by the SET PORT BIOSn command, where n is 1, 2, 3, or 4, as appropriate for the LAN software. To work properly the LAN must support the same use of registers as the system Bios. Several X.25 and TCP/IP packages have been operated successfully with the SET PORT BIOSn command. Since this channel appears to Kermit as a simple software level serial port, Kermit provides neither interrupt driven i/o nor LAN session support.

Kermit can access files on the LAN file server via DOS even while using the LAN as a communications medium. Network administrators should note this point because a user operating Kermit in Server mode can allow his or her file server directories to be available to other network users also running Kermit, without additional security checking of the other users. The network drives visible to the Server Kermit can become devices available for Kermit-to-Kermit file transfers, etc, unless the DISABLE command is used to confine access to the current disk and directory. A corollary is when files are accessible to DOS commands they can become public.

1.18.2. Use of Kermit-MS with External Device Drivers

It is often desirable to supplement or modify the behavior of a DOS program by loading it with special external device drivers. These drivers may operate at either the DOS or BIOS level. When Kermit-MS accesses the BIOS directly, DOS-level drivers are ineffective. When Kermit accesses the hardware directly, both the DOS and the BIOS level drivers are bypassed. Kermit-MS provides several mechanisms to allow these external drivers to operate as intended.

Here are a few examples:

- IBM's `ANSI.SYS` console driver operates at the DOS level. It allows the major IBM PC keys to be redefined, and also interprets selected ANSI-format escape sequences for screen control. It works fine at Kermit-MS command level, except SHOW KEY does not recognize strings assigned to keys via `ANSI.SYS`, and fine at CONNECT level. To use `ANSI.SYS` at CONNECT level, issue the Kermit-MS commands SET KEY OFF (to read keys via DOS) and SET TERMINAL NONE (to display characters through DOS).

- Blind people often have speaking or Braille machines attached to their PCs. DOS-level device drivers are generally used to redirect screen output to these devices, which works OK at DOS or MS-Kermit command level. SET TERMINAL NONE will allow this redirection to take place during CONNECT. But these devices also need to have the computer's output appear as a coherent stream of text, so users should also take care to inform the remote host to format its output for a "dumb" or hardcopy terminal. In addition, Kermit-MS' normal file transfer display does not mesh well with these devices, but that can be remedied using SET DISPLAY SERIAL.

- People with motor impairments may be using special keyboard replacements supported by DOS-level device drivers. As with `ANSI.SYS`, Kermit-MS may be directed to use such keyboard drivers with the command SET KEY OFF.

- Other keyboard drivers are available that work, like Kermit-MS, at BIOS level. Examples include ProKey and SuperKey. These may be used at DOS or Kermit-MS command level as well as during CONNECT.

- Conceivably, drivers exist that allow DOS communication programs to emulate terminals other than ANSI. You should be able to use them, if they exist, in conjunction with Kermit-MS by telling Kermit to SET TERMINAL NONE, but the speed may not be high because of the intervening DOS calls.

1.18.3. Kermit-MS/IBM Serial Port Information

Kermit-MS for IBM PC's and compatibles does testing of serial ports before use. This section describes those tests so users may understand what Kermit does.

When a serial port is selected by the SET PORT COMx command Kermit looks at low memory addresses in segment 40H assigned to hold the base address of each COMx port; COM1 is in word 40:0H, COM2 is in word 40:2H, and so on. If the value in the appropriate word is binary zero then Kermit declares the port to be unavailable. Otherwise, Kermit runs read-only (i.e., safe) tests at the base address to validate the presence of an official 8250 UART chip. If the tests fail Kermit indicates it will do i/o through the slow Bios pathway; some PC clones need to work this way even though the Bios has speed problems even at 1200 baud. Otherwise, interrupt driven i/o will be done through the 8250 UART (that is, very fast).

There is a special case when a communications board is present, set for COM2, but a normal COM1 serial port is not. Kermit detects this situation.

Many machines now have more than two serial ports, but until recently there has been no standard about addresses for COM3 and COM4. PC DOS 3.30 does not assign them either because it is really a problem of the system ROM Bios boot code run when the power is turned on. However, Kermit will use COM3 and/or COM4 if the base address of a port is placed in low memory words 40:4H (COM3) or 40:6H (COM4); the tests described above are then carried out. One restriction is that the Interrupt ReQuest number (IRQ in the serial port board manual) must be either IRQ4 or IRQ3. Kermit attempts to locate which line is correct with a short test. If the test is not successful it uses the IRQ4 for COM3 (and for COM1) and IRQ3 for COM4 (and for COM2) on the PC/AT, and on the PS/2 it uses IRQ3 for COM2, COM3, and COM4. Check the board and its manual. DOS utility DEBUG can be used to create a short program to insert the board's addresses into the segment 40H memory locations; a sample program is given below.

Serial Port	Address	IRQ Line	Conventions
COM1	03F8H	4	IBM standard
COM2	02F8H	3	IBM standard
COM3	?	4 (3 for PS/2)	Board
COM4	?	3	Board

Table 1-13: IBM PC/XT/AT Serial Port Numbers

The addresses shown as query marks are to be found in the board's reference manual; values such as 2E8H and 2E0H would be common. However, there is no standard for anything to do with COM3 and COM4 on non-PS/2's.

Assuming that you have selected an address in harmony with the rest of the system (good luck on that part), set the board's switches or jumpers, and use DEBUG to insert the address(es) in segment 40H memory. The example below creates a small program named `SETCOM3.COM` to put address 02E8H into the memory word 40:04H for COM3 and writes the program to drive A. (Disregard the xxxx items below):

```
A> DEBUG                          don't type these comments
-n a:setcom3.com                  sets name of output file
-a                                assemble command
xxxx:100 mov ax,40                value 40h
xxxx:103 mov es,ax                put it into register es
xxxx:105 mov ah,02                the 02 part of 02E8H
xxxx:107 mov al,e8                the E8 part of same
xxxx:109 es:
xxxx:10A mov [4],ax               store in 40:4 for com3 ([6] for com4)
xxxx:10D int 20                   return to DOS
xxxx:10F                          blank line to end assemble mode
-r cx                             show contents of register cx
CX 0000
: 0f                              set register cx to write 0fh bytes
-w                                write material to the disk file
-q                                quit debug
A> DEBUG setcom3.com
-u                                unassemble to see if all is well
-q                                quit debug
```

Note, for COM4, use `[6]` above rather than `[4]`, and of course employ your board's port address in place of 02E8H (check the manual). Finally, try it:

```
A> setcom3                        run the program
A> DEBUG                          now see what's down there
-d 40:00                          display bytes in seg 40H

       ( Shows many bytes.  See yours?  Good. )

-q
A>
```

A small side effect noted in practice is the first time the extra port is used there may be garbage from it. Just return to the Kermit prompt and try again, if necessary SET PORT to the other COM lines momentarily, all should be well the second time.

More technical comments, for those with an interest. When Kermit finishes with a port it disables interrupts for that serial port and returns the IRQ signal line to its state found when Kermit started since many devices can share the same Interrupt ReQuest line but only one device at a time can be active on it. If you find that transmissions are good but there is no reception then another device has stolen the IRQ; disable it or find a guru. Kermit will work with non-standard addresses for COM1 and COM2 but the IRQ's must be as in the table above. Accessing a non-existent port produces a message and all communications are discarded safely in the bit bucket.

1.18.4. CTTY COMx for IBM Machines

The DOS command CTTY COMx redirects the standard input and output from the keyboard and screen, respectively, to the indicated communications channel. If a Kermit Server is operated this way, "through the back port", then both DOS and Kermit can access the port hardware simultaneously; a deadlock develops on IBM machines. The items below refer to only the IBM version of Kermit-MS.

Kermit-MS/IBM version successfully resolves the deadlock in the following manner. When Kermit requires the serial port it also attaches itself to Interrupt 16H, the Bios RS232 serial port routine. Code within Kermit receives the DOS serial port requests via Interrupt 14H and either passes the request to the Bios if the COM line is not that used by Kermit or it handles the request internally for conflicting situations. When the same port is used by both DOS and Kermit, Kermit discards DOS output material (typically a prompt, but could be the dreaded Abort, Retry, Ignore message) and returns a success code to DOS, it returns an ascii Backspace code to DOS read requests (this is a key item to keep DOS complacent while Kermit communicates), and it returns reasonable status for modem status. The interception ceases when Kermit releases the port, such as when the Kermit prompt is displayed, and this lets DOS converse out the serial port.

It is worth restating that a large number of programs bypass DOS to achieve higher performance. When such programs are started through the back door they may still require input from the real keyboard and will hang, waiting for it. There is nothing to do about this situation except a) don't let it happen, b) contact the local operator to push some keys.

1.18.5. Screen Sizes and the EGA Board, IBM Versions

Support has been included for Enhanced Graphics Adapter (EGA) video display boards which can be configured for other than the standard 80 columns by 25 lines, say 132 columns or 43 lines or other. Several boards, the Tseng Labs EVA (also Orchid Designer) board with the 132 column kit installed, the ATI EGA Wonder, the Video 7 Deluxe and VGA, and the Everex EV-659 (ega) and EV-673 (vga), can be controlled directly by Kermit for 80/132 column changes. Other boards need to be placed in the desired display mode by the user. Kermit then adapts to the settings if the board obeys standard rules for using the Bios EGA memory areas in segment 40H. The Video-7 boards have been used successfully in all screen sizes, including 132 columns by 43 lines, with an NEC Multisync monitor.

The IBM EGA board has several noteworthy bugs which are now standards. One is the cursor dots are not always on the correct scan lines when the number of screen lines is other than 25. Kermit-MS attempts to compensate for this attribute. Screen roll back space is fixed in size so there are fewer pages for more dense screens; standard screens use an internal buffer, non-standard screens use a buffer plus video page 1. `ANSI.SYS` is hard coded for 25 line displays so all DOS i/o will eventually overwrite itself on line 25; the emulator does not use DOS i/o. Commercial replacements for `ANSI.SYS` should be able to use all screen lines.

Screen dumps work correctly if done with Kermit commands. DOS PrintScreen may or may not, depending on your EGA board. Graphics dumps are not managed by Kermit.

When the VT102 receives escape sequences to change between 80 and 132 column modes the screen is reset and the ATI EGA Wonder, or Everex EV-659 (EGA) or EV-673 (vga), Tseng Labs Multipak (and Orchid Designer), or

Video 7 Vega or VGA board is asked to change modes (but only if that board is present); other display adapters are left in their current state. Users of Tseng boards must run the Tseng BIGSCR /R:25 program before starting Kermit. The right margin is enforced strongly so a board in 132 column mode will not display material to the right of column 80 if the emulator is in 80 column mode. Similarly, material to the right of column 80 is not preserved in the emulator if the display adapter is operating in 80 column mode; real VT102s keep that invisible material in hardware memory whereas the emulator does not.

Reference is made to line 25 in the emulator; this is normally the status/mode line in Kermit. Real VT102's have only 24 line displays. If the display adapter is set for a different number of lines per screen then the 25th line is interpreted to mean the bottom display adapter line, such as line 43. Should the host access the status/mode line then the line is declared to be disabled (same as SET MODE OFF) so that Kermit's own status information does not overwrite the host's when the screen is restored. Toggling a disabled mode line has no effect; only SET MODE ON will enable it again. The Heath-19 terminal has the unusual feature that disabling the mode line `(ESC y 1)` also clears it.

1.18.6. Kermit-MS/IBM Printer Control

The IBM PC MS-Kermit VT102 terminal emulator also supports full transparent printing of 8-bit binary bytes. The escape sequence ''`ESC [ 5 i`'' turns on transparent printing, in which all further 8-bit characters are sent directly to the printer, bypassing the SET TRANSLATION INPUT filter, and are not shown on the screen. Escape sequence ''`ESC [ 4 i`'' turns off transparent printing and the escape sequence is not sent to the printer. Non-transparent printing is controlled by the ''`ESC [ ? 5 i`'' and ''`ESC [ ? 4 i`'' sequences. Such printing simply duplicates text intended for the screen, excluding escape sequences. The text also appears on the screen.

Kermit-MS accesses the system printer through DOS calls several ways; neither the Bios nor the hardware are used. Files directed to the printer by the SET DESTINATION PRINTER command are written by opening a file with the name PRN (DOS's name for the system printer) and writing to it the same as to a disk file; DOS provides limited buffering. LOGging to device PRN works the same way, as can be noticed by the last line or so not being printed until the log file is CLOSED. DOS is used again while emulating a terminal in CONNECT mode. If the VT102 emulator found in the IBM PC is used for transparent or Controller printing, single characters are written to DOS file handle 4, the DOS standard print device. If the screen is echoed to the printer via the typical Control PrtSc key combination, or equivalent, single characters are written by the DOS function 05H Printer Output call. In both cases of terminal emulation the printer's ready status is found by the DOS IOCTL 44H call. Only the Control PrtSc case results in the PRN message being displayed on the status line. Finally, the classical IBM PC Shift PrtSc command to copy the whole screen to the printer is unknown to Kermit because the system Bios traps the key combination and does not tell Kermit about it. If the Control P command is given to DOS before Kermit starts then again characters are echoed by the system Bios without Kermit's knowledge; this situation can result in lost characters.

Print spoolers generally operate by being told an existing filename and then in the background they steal cpu cycles to read from disk and write to the printer. The DOS PRINT command invokes such a spooler. Although an active Kermit does not feed these software programs directly the spooler and Kermit can compete for cpu cycles and characters can be lost. If a non-DOS resident program intercepts characters destined for the printer device and spools them Kermit does not know about it and similar competion can occur.

During file transfers printing is carefully sequenced to occur only when the local Kermit is in control of the communications line so that a small pause will not result in missing characters arriving at the serial port. When terminal emulation is active then printing competes for cpu time with the serial port routines. Generally, the serial port wins such contests if the port is interrupt driven (Generic Kermit is not interrupt driven, so beware). However, the printing itself can use enough cpu cycles to delay processing of characters to the screen and eventually the receive buffer of the serial port fills to the high water mark and an XOFF flow control character is sent to the host to suspend further transmissions until we send an XON. If FLOW is NONE then expect lost characters at the serial port. Experience with ordinary IBM PC's through 80386 machines at very high baud rates indicates no characters are lost when FLOW is XON/XOFF. However, it is possible on some machines for the printer to have priority over the serial port, and hence to have lost characters, especially if a Terminate Stay Resident program intercepts characters destined for the printer and keeps interrupts turned off too long.

Index

Table of Contents

List of Figures

List of Tables