CP/M-80 KERMIT VERSION 4.11 USER GUIDE

C. Gianone

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

April 23, 1991

1. CP/M-80 KERMIT

Program: Mike Freeman, Bonneville Power Administration, Vancouver, WA, USA, with contributions from many others.

Language: 8080 Assembler, LASM, M80, or MAC80

Version: `4.11`

Date: April 1, 1991

Documentation: Christine Gianone, Columbia University, with contributions from many others.

KERMIT-80 Capabilities At A Glance:

Local operation:	Yes
Remote operation:	Partial, Auto-receive only
Login scipts:	Yes, limited
Transfer text files:	Yes
Transfer binary files:	Yes
Wildcard send:	Yes
File transfer interruption:	Yes
Filename collision avoidance:	Yes
Can time out:	Yes
8th-bit prefixing:	Yes
Repeat count prefixing:	No
Alternate block checks:	Yes
Terminal emulation:	Yes, VT52 and others
Communication settings:	Yes
Support for dial-out modems:	No
Transmit BREAK:	Yes; most versions
IBM communication:	Yes
Transaction logging:	No
Debug logging:	No
Session logging:	Yes
Raw file transmit:	Yes
Act as server:	No
Talk to server:	Yes
Advanced commands for servers:	Yes
Command/init files:	Yes
Command macros:	No
Local file management:	Yes
Handle file attributes:	No
Long packets:	No
International Character Sets:	No
Sliding Windows:	No
Printer control:	Yes, limited

1.1. Credits

CP/M Kermit is the first of all the Kermit programs. It was originally written by Bill Catchings of Columbia University in 1981. Over the years, contributions have been added by many people, including Charles Carvalho (ACC), Bernie Eiben (DEC), Nick Bush (Stevens Institute of Technology), John Bray (University of Tennessee), Bruce Tanner (Cerritos College), Greg Small (University of California at Berkeley), Kimmo Laaksonen (Helskini University of Technology), Bertil Schou (Loughborough University), Jon Warbrick (Plymouth Polytechnic University), Brian Robertson (Aberdeen University), A.J. Cole (Leeds University), John Shearwood (Birmingham University), Tony Addyman (Salford University), Godfrey Nix and Martin Carter (Nottingham University), Ian

Young (Edinburgh University), Chris Miles (Manchester University), Richard Russell, Dave Roberts, and many, many others.

Version 4.11 is the work of Mike Freeman of the Bonneville Power Administration in Vancouver, WA, USA, with assistance from Russell Lang of Monash University in Australia, Jay S Rouman of Mt Pleasant MI, and others.

1.2. What's New

Features added since version 4.09 include:

- SET COLLISION {BACKUP/DISCARD/OVERWRITE/RENAME}
- SET INCOMPLETE-FILES {DISCARD/KEEP}
- Many REMOTE commands, including some REMOTE SET commands
- RENAME command to rename CP/M files from within Kermit-80
- SET RECEIVE/SEND PACKET-LENGTH nn (nn <= 94)
- SET AUTORECEIVE ON now implies that Kermit-80 ALWAYS tries to receive more files when a RECEIVE transaction has completed. The user can cancel with ^C.
- QUIT is now a synonym for EXIT.
- STAY is now a synonym for SET NO-EXIT.
- CONNECT, RECEIVE and SEND may be abbreviated to C, R and S, respectively.
- Cancellation of TAKE, TYPE, and PRINT commands from the keyboard.
- Many bug fixes.
- Kermit-80 Version 4.11 now supports the Microbee family of computers (56K, 64K, 128K and 256K) manufactured by Microbee Systems, Ltd, of Australia.
- Kermit-80 now supports the Ampro Little Board system.

1.3. Overview of Kermit Operation

Use the SET command to establish necessary communication parameters like SPEED and PARITY. Use the CONNECT to establish a terminal connection to the remote computer. If you are dialing out with a modem, type the necessary dialing commands to the modem first. The dialing process can be automated to some extent using a TAKE command file containing INPUT, OUTPUT, and PAUSE commands. Then log in to the remote computer or service and conduct a session.

To transfer a text file, start the Kermit program on the remote computer and tell it to SEND the desired file (if uploading) or to RECEIVE (if downloading). "Escape back" to CP/M Kermit, usually by typing Ctrl-] (hold down the Control key and press the right bracket key) and then type the letter C. At the CP/M Kermit prompt type RECEIVE (if you gave a SEND command to the remote Kermit) or SEND *filename* (if you gave a receive command to the remote Kermit).

To transfer a binary file, give the command SET FILE TYPE BINARY to the remote Kermit and SET FILE-MODE BINARY to CP/M Kermit before issuing any SEND or RECEIVE commands.

Multiple files of the same type (text or binary) can be transferred in a single operation using "wildcard notation" (including special characters like asterisk in the filename).

When file transfer is complete, CONNECT back to the remote computer, use the EXIT command to exit from the remote Kermit program, finish your work on the remote computer, log out from it, escape back to CP/M Kermit again, and EXIT from CP/M Kermit.

The remote Kermit may also be put into "server mode" to simplify these operations. Give the SERVER command to the remote Kermit, escape back to CP/M Kermit, and then issue SEND commands to send files (upload), GET *filename* commands to receive (download) files, REMOTE commands to request various other services (like directory listings) from the remote Kermit. When you are done, give a BYE command to terminate your remote session, or a FINISH command to tell the remote Kermit to return to its prompt so you can CONNECT back and conduct further business.

That's all there is to it.

1.4. Summary of CP/M

There are essentially two versions of CP/M - Versions 2.2 and 3.0 (sometimes also called CP/M PLUS.)

CP/M-80 Version `2.2` is run in a single 64 Kbyte "page", usually the largest amount of memory on Z80 or 8080 systems. The BIOS (Basic input/output system), BDOS (Basic Disk Operating System) and CCP (Command console processor) all share memory with any transient program the user may wish to run. Some basic commands are available through the CCP, like DIR, ERA etc,while others are loaded from disk into the transient program area and run as a program, like PIP or STAT.

CP/M Version `3.0` (or CP/M PLUS) effectively removes the requirement of having the CCP and BDOS along with a chunk of the BIOS code being resident in the single 64k byte page of memory. This allows even more space for programs in the TPA, but still a little less than the maximum of 64k. It is substantially different from CP/M version `2.2`, with lots of added features. Kermit-80 uses very few additional version `3.0` features, and only where absolutely necessary.

CP/M file specifications are of the form `DEV:XXXXXXXX.YYY`, where

`DEV:`	is a *device name*, normally the A: or B: floppy. If omitted, the device name defaults to your connected diskette.
`XXXXXXXX`	is a *filename* of up to 8 characters.
`YYY`	is the *file type*, up to 3 characters.

File names and file types may contain letters, digits, and some special characters, including dash, dollar sign, and underscore, but no imbedded spaces. Upper and lower case letters are equivalent.

"Wildcard" file-group specifications are permitted in file names and file types (but not device names) within certain contexts; a "`*`" matches a whole field, a "`?`" matches a single character, including space. Examples: "`*.F??`" specifies all files whose *types* start with F and are 1, 2, or 3 characters long; "`F?.*`" specifies all files whose names start with F and are no more than two characters long (before the trailing spaces).

The five CP/M commands are:

DIR *file*	Lists the the names of the specified files. The default file specification is "*.*". Example: "`DIR B:*.FOR`".
ERA *file*	Erases (deletes) the specified file(s); wildcards allowed.
REN *new old*	Changes the name of a file from *old* to *new*, e.g. "`REN NEW.FOR=OLD.FOR`".
SAVE	Saves the specified number of memory blocks into a file. (Not on CP/M Plus systems)
TYPE *file*	Types the specified file on the screen, e.g. "`TYPE FOO.TXT`".

The most important programs are:

STAT	Gives statistics on disk usage; sets and displays IOBYTE. (Not on CP/M Plus systems)
PIP	Peripheral Interchange Program. Copies files. In response to the "`*`" prompt, give a command of the form `disk:outfile=disk:infile` Wildcards ("`*`" for a whole field or "`?`" for a letter) can be used. Examples: "`A:=B:*.*`" to copy a whole disk, "`A:=B:*.FOR`" to copy all the Fortran programs from disk B to disk A. If the disk specification is omitted, your "connected" disk is assumed. Command line arguments are also accepted, e.g. "`PIP A:=B:*.*`".

There are equivalent commands for CP/M Version 3.0, but are not loaded into memory in the same way as for CP/M Version 2.2. For further information on CP/M, consult your microcomputer manual or a CP/M handbook.

1.5. Kermit-80 Description

Since Kermit-80 runs on a standalone micro, it is always in control of the screen -- it is always in "*local* mode". It includes a terminal emulator for establishing a connection to a remote computer or service, and during file transfer, it keeps the screen updated with the file name and the packet number, whether sending or receiving.

Kermit-80 is capable of an imprecise or "fuzzy" timeout on an input request, and can break deadlocks automatically. In most cases, this is not important, because the Kermit program on the other side is most likely able to handle the timeouts. The timeouts done by Kermit-80 are fuzzy because they depend on the speed of the processor and other factors that can vary from system to system.

If, despite the timeout capability, the transmission appears to be stuck (and you can tell that this has happened if the screen fails to change for a while) you can type carriage return to have the micro do what it would have done on a timeout, namely NAK the expected packet to cause theforeign host to send it again (or, if the micro is sending, to retransmit the last packet). Micro/micro or micro/IBM-mainframe transfers could require this kind of manual intervention.

File transfers may be interrupted in several ways.

Control-C	This will return you to Kermit-80 command level immediately, so that you can connect back to the remote system, or take any other desired action.
Control-X	When sending a file, this will terminate the sending of the current file with a signal to the KERMIT on the other side to discard what it got so far. If there are more files to be sent, KERMIT-80 will go on to the next one. When receiving a file, KERMIT-80 will send a signal to the remote KERMIT to stop sending this file. If the remote KERMIT understands this signal (not all implementations of KERMIT do), it will comply, otherwise the file will keep coming. In any case, the remote KERMIT will go on to the next file in the group, if any.
Control-Z	Like Control-X, except if a file group is being transmitted, this will stop the transmission of the entire group. If only a single file is being transmitted, it works exactly like Control-X.
Carriage Return	If you type a carriage return Kermit-80 will resend the current packet. You may do this repeatedly, up to the packet retry limit (somewhere between 5 and 16 times) for a particular packet.

Kermit-80 Commands

Kermit-80 is an interactive program. It issues a prompt, you type a command. The process repeats until you give the EXIT command to leave the program.

Commands consist of keywords, filenames, and numbers. Keywords may be abbreviated to minumum unique length. "?" may be typed to request a menu of the available options for the current field at any point in a command. ESC may be typed at any point in a command to fill out the current keyword or filename; if sufficient characters have not been typed to identify the current field uniquely, Kermit-80 will sound a beep and allow you to continue from that point. Here are Kermit-80's commands:

BREAK Send a BREAK condition to the remote computer. This is only possible if your system is capable of sending breaks. It is intended to be used with PAUSE, OUTPUT, etc and the TAKE command to do wierd and wonderful things, like automatic logging on to a remote host.

BYE When talking to a remote Kermit Server, this command shuts down the server and logs it out, and also exits from Kermit-80 to CP/M command level.

CONNECT

Establish a terminal connection to the computer, service, or device that is connected to the serial port, i.e. pass all typein to the serial port and display all input from the serial port on the screen. Also, emulate a DEC VT52 to allow cursor control, screen clearing, etc., if VT52-EMULATION is ON (see below), in which case you should also set your terminal type on the remote host to VT52. (Some versions emulate other terminals.) The CONNECT command may be abbreviated by the single letter C.

Warning: VT52 emulation is only successful if your system or its attached terminal can do the same sort of functions as a genuine VT52. Things to beware of are cursor addressing, clear to end of page and end of line, clear screen, home cursor, and clear-and-home functions. The useability of VT52 emulation depends entirely on how many of the VT52 functions can be emulated by your micro or terminal.

The escape character differs from micro to micro; when you issue the CONNECT command, the micro will print a message telling you how to get back. The escape sequence is generally an uncommonly-used control character, like CTRL-backslash or CTRL-rightbracket, followed by a single letter "command":

C	Close Connection, return to `Kermit-80>` command level.
S	Display Status of connection, but maintain remote connection.
`?`	List available single-character commands.
0	(zero) Send a null (0) character.
B	Send a BREAK signal. Most systems provide this function.
D	Drop the line. Used on the Apple with modem. Automatically closes the connection after dropping the line. The TORCH system acknowledges this command but does nothing.
P	Toggle printer on or off. Allows you to copy whatever goes to the screen to the printer.
S	Temporarily suspend logging to the log file.
Q	Restart logging to the log file
`^]`	(or whatever - a second copy of the escape character) Send the escape character itself to the remote host.

COPY *source destination*

Copy a named file to another file, either on the same drive or another drive.

DIRECTORY

This provides a directory listing of the specified files. If no files are specified, all files on the default disk are listed. File sizes, in K, are included. You may interrupt the listing at any time by typing any character. The listing (even if interrupted) concludes with a display of the amount of free storage left on the disk. You can inhibit the display of file sizes by SET DIRECTORY OFF.

ERASE *filespec*

This executes the CP/M ERA command on the specified file(s). The names of the files being erased are not displayed.

EXIT Quit back to CP/M. The return is made by a JMP 0 (Warmstart). QUIT is a synonym for EXIT.

FINISH Like LOGOUT, but shuts down the remote server without logging it out. Leaves you at Kermit-80 command level; subsequent CONNECT commands will put you back at host system command level.

GET *filespec* [*local_filespec*]

When Kermit-80 is talking to a Kermit Server on the host, you should use the GET command to request the server to send files to you, for example:

```
get hlp:k*.hlp
```

You may specify a local filename if you want to save the remote file under a different filename. Limitation: If you request an alternate block check type using the SET BLOCK command, the GET command will not communicate it to the remote server. If you want to have type 2 or 3 block checks done when getting files from the server, you have to issue the appropriate SET BLOCK command to the remote KERMIT before putting it in server mode.

HELP List all these commands, with a short description on what the commands do. A question mark will do the same. If you have already typed a command but do not know what the parameters are, type a space (to indicate the end of the command) and a question mark. You will be informed of what Kermit can expect at that stage.

INPUT *seconds text*

Setup a text line and time delay for your CP/M system to expect from the host, then wait up to the given number of seconds (approximately) for text to be sent to your CP/M-80 system.

LOG *filespec*

When CONNECTed to a foreign host as a terminal, log the terminal session to the specified diskette file. This functionality depends to some extent on the remote host's ability to do XON/XOFF flow control, and does not guarantee a complete transcript (after all, that's what the KERMIT protocol is for). The log file is closed when the connection is closed by typing the escape character followed by the single-character command "C".

It is possible to temporarily suspend logging during connect state. Typing an escape sequence can turn file logging on (<escape-character> R for Resume) or off (<escape-character> Q for quiet).

Re-entering connect state will re-open the previously opened log file and append to that file.

LOGOUT

Like BYE, but leaves you at Kermit-80 command level.

OUTPUT *text*

Send the text to the remote computer as if you had typed it.

PAUSE *seconds*

If this command is issued your CP/M system will wait a while before proceeding with another command. This is intended for use in TAKE commands, where you may want to pause for a while before proceeding with the rest of the TAKE file. The actual delay is very variable between systems, and values should be determined on a trial and error basis.

PRINT Print a file to the console and printer. Output to the printer is buffered by the Kermit-maintained printer buffer. This routine is identical to TYPE but characters are echoed to the printer as well as to the screen. Suspending and canceling output is as described in TYPE.

QUIT Synonym for EXIT.

RECEIVE *filespec*

Receive file(s) from the remote Kermit, and save them under the names provided in the file headers supplied by the remote host. If a local filespec is given, the file is saved under the given filename. If the names aren't legal, use as many legal characters from the name as possible (see the description of SET FILE-WARNING below). If there's a conflict, and FILE-WARNING is ON, warn the user and try to build a unique name for the file by adding "&" characters to the name. RECEIVE can be abbreviated to the single letter R.

REMOTE *command*

Send a command to a remote Kermit server. The results are sent back to your CP/M screen. When two arguments are required and specify less than two in the command, you will be prompted for the missing arguments. REMOTE commands include:

REMOTE CD [directory]

Ask the remote server to change its default directory. If no directory is specified, the server changes to its login directory.

REMOTE COPY file1 file2
Ask the remote server to copy file1 to file2.

REMOTE RENAME file1 file2
Ask the remote server to rename file1 to file2.

REMOTE DELETE filespec
Ask the remote server to delete the named file or files.

REMOTE DIRECTORY [filespec]
Ask the remote server to display a directory listing of the given files or, if the filespec is omitted, all the files in the current device or directory.

REMOTE DISK-USAGE
Ask the remote server to display information about its disk usage (such as free or used space).

REMOTE ERASE filespec
Same as REMOTE DELETE.

REMOTE FINISH
Same as FINISH.

REMOTE HELP
Ask the remote server to display a list of the commands it can respond to.

REMOTE HOST command
Ask the remote server to have its operating system execute the given command.

REMOTE KERMIT command
Ask the remote server to execute the given Kermit command, given in the server Kermit's command syntax.

REMOTE LOGIN user password
Log in to a remote Kermit server which has been set up to require a username and password.

REMOTE MESSAGE text
Send the text to the remote server for display on its screen (useful with MS-DOS Kermit servers).

REMOTE SET parameter value
Ask the remote server to set the given parameter to the given value, for example REMOTE SET FILE TYPE BINARY. Type REMOTE SET ? to see a list of the REMOTE SET options.

REMOTE SPACE
Same as REMOTE DISK-USAGE.

REMOTE STATUS
Ask the remote server to provide a status report.

REMOTE TYPE file
Ask the remote server to display the named file on the micro's screen.

REMOTE WHO [user]
Ask the remote server for a list of users who are logged in, or if a user is specified, for a report on the named user.

RENAME file1 file2
Rename local CP/M file1 to file2.

SEND *filespec*
Send file(s) specified by *filespec* to the remote Kermit. The *filespec* may contain CP/M wildcards. SEND may be abbreviated to the single letter S.

SET *parameter* [*value*]
Set the specified parameter to the specified value. Possible parameter settings:

AUTORECEIVE
ON (or OFF). Allows several files to be received without having to type RECEIVE on the

receiving machine. The routine simply looks for activity on the serial line, and if so fudges a RECEIVE command. The packet sent by the sender will be lost.

BLOCK-CHECK-TYPE *option*
The options are:

1-CHARACTER-CHECKSUM
Normal, default, standard 6-bit checksum.
2-CHARACTER-CHECKSUM
A 12-bit checksum encoded as two characters.
3-CHARACTER-CRC-CCITT
A 16-bit CCITT-format Cyclic Redundancy Check, encoded as 3 characters.

BUFFER-SIZE *value*
This allows you to set a buffer size during transfer of data. On some systems it takes so long that the remote end times out while the local system is reading or writing to disk. The size is the number of 128 disk sectors (nominal) and can be from 1 (128 bytes) to 64 (8 kbytes).

CP/M-80 filenames will still be mapped to uppercase characters.

COLLISION value
What to do when a file arrives that has the same name as an existing file. BACKUP means to rename the existing file. DISCARD means to discard and reject the incoming file. OVERWRITE means to overwrite the existing file. RENAME means to rename the existing file.

DEBUG ON (or OFF). Enables/disables displaying of packets on the screen during file transfer. Not performed if the QUIET option has been set for the terminal (SET TERMINAL QUIET)

DEFAULT-DISK *drive letter*
This allows you to set the default disk as source and destination of file transfers. In addition, issuing this command causes you to switch to the specified disk and log it in, write-enabled. The colon must be included in the disk name (A:). The selected disk appears in your KERMIT-80 prompt, for instance

```
Kermit-80 14A:>
```

DIRECTORY-FILE-SIZE ON (or OFF).

By setting DIRECTORY-FILE-SIZE OFF you can get an abreviated listing of your disk drive. File sizes are not calculated, and five files are shown on a line. Setting this option ON will show file sizes of each file.

Both options will list the free space remaining.

ESCAPE Change the escape character for virtual terminal connections. Kermit-80 will prompt you for the new escape character, which you enter literally.

FILE-MODE *option*
Tells KERMIT-80 what kind of file it is sending, so that KERMIT can correctly determine the end of the file. SET FILE BINARY means to send all the 128-byte blocks (ie logical CP/M sectors) of the file, including the last block in its entirety; SET FILE ASCII is used for text files, and transmission stops when the first Control-Z is encountered anywhere in the file (this is the CP/M convention for marking the end of a text file).

SET FILE-MODE DEFAULT tells Kermit to attempt to determine the file type by examining the file being transmitted. If a Control-Z appears before the last block of the file, it is assumed to be BINARY; if, when the first Control-Z is encountered, the remainder of the file contains only control-Z's, it is assumed to be a text file. Unfortunately, not all programs fill the remainder of the last record of a text file with Control-Z's, so this algorithm is not always successful.

If binary transmission is used on a text file, or a compressed file (eg a .DQC file) some extraneous characters (up to 127 of them) may appear at the end of the file on the target system.

If ASCII transmission is used on a binary file, any 8th bits set will be stripped and a warning sent to the console. When the first control-Z is encountered, the file is assumed to be at the end, even if it is not.

FLOW-CONTROL ON (or OFF)
: Sets XON/XOFF flow control on or off. If set ON the host is expected to respond to an XOFF or XON sent by Kermit-80. If set off, no flow control is assumed and any XON/XOFF is ignored.

IBM ON (or OFF)
: Allow the transfer of files to and from an IBM mainframe computer. This makes Kermit-80 wait for the IBM turnaround character (XON), ignore parity on input, add appropriate parity to output, and use local echoing during CONNECT. As distributed, KERMIT-80 uses MARK parity for IBM communication. If you don’t give this command, IBM mode is OFF. Since IBM VM/CMS KERMIT does not have timeout capability, SET IBM ON also turns on the "fuzzy timer" automatically.

LOCAL-ECHO ON (or OFF)
: When you CONNECT to a remote host, you must set LOCAL-ECHO ON if the host is half duplex, OFF if full duplex. OFF by default.

LOGGING ON (or OFF)
: Cease or resume logging whenever connect mode is entered. This is really only applicable after a LOG command is no longer required.

NO-EXIT
: This command is applicable only for Kermit initiated with a command tail. For example, if Kermit was initiated by:

```
KERMIT ;SEND HELLO;NO-EXIT
```

Kermit would first seek out and execute the KERMIT.INI file (if present), then send file HELLO to a remote system. Usually Kermit would exit back to CP/M, but NO-EXIT over-rides this. STAY is a synonym for NO-EXIT.

Note the leading semicolon. This clears leading spaces from the first command.

OUTPUT *text-line*
: Send a line of text to the remote computer (or modem). This simply copies the string to the correct line, and assumes all appropriate parameters have been set to be used, e.g. speed, parity etc. It is intended for use in TAKE command files.

PARITY *option*
: Sets parity for outgoing characters to one of the following: NONE, SPACE, MARK, EVEN, or ODD. On input, if parity is NONE, then the 8th bit is kept (as data), otherwise it is stripped and ignored. The parity setting applies to both terminal connection and file transfer. If you set parity to anything other than none, KERMIT-80 will attempt to use "8th bit prefixing" to transfer binary files. If the other KERMIT is also capable of 8th bit prefixing, then binary files can be transferred successfully; if not, the 8th bit of each data byte will be lost (you will see a warning on your screen if this happens).

PORT *port name*
: Allows you to switch between different communication ports. This command is not available on all systems. Type `SET PORT ?` for a list of valid options for your system. (Note: If your system does not support several ports, this command will return a "Not implemented" error if you try to set a port.)

PRINTER
: ON (or OFF)
: Turns copying of CONNECT session to printer on and off. It is also possible to toggle the printer on/off from the connect state, by typing <escape character> followed by P.

RECEIVE *parameter* [*value*]
: Set a RECEIVE parameter.

PAD-CHAR
: Set the PAD character to use while receiving files. Currently a dummy, as for SET SEND PAD-CHAR.

PADDING [value]

Set the number of PAD characters to use while receiving files. Same as SET SEND PADDING.

START-OF-PACKET [value]

Set the default start of Packet character for receiving files. Apply the same rules and considerations as for SET SEND START-OF-PACKET.

PACKET-LENGTH number

Tell the other Kermit the longest packet length CP/M Kermit is willing to receive during file transfer. The maximum length is 94, which is also the default length.

SEND *parameter* [*value*]

Set a SEND parameter.

PAD-CHAR

Set the Pad character to be used while sending files. It is currently a dummy entry, and does not do anything.

PADDING [value]

Set the number of PAD-CHARS to be used while sending files. This too does nothing.

START-OF-PACKET

Set the default start of packet character to another character than control-A. This may be necessary on systems (including intervening networks) that trap control-A characters. Choose a control character not otherwise used, ie not carriage return (13D, ODH), line feed (10D, OAN), tabs (09D, 09H), backspace (08H), and bell (07H) or any other used between you and your remote system.

SPEED *value*

Change the baud rate of the communications port. This command only works on some systems. *value* is the numeric baud rate (300, 9600, etc.) desired. Type SET SPEED followed by a question mark for a list of supported baud rates. On systems that do not support this command, you must set the port baud rate from CP/M or other setup mechanism outside of KERMIT-80.

TACTRAP

Set the TAC intercept character. If you are attached to a TAC it will swallow the intercept character (commercial AT sign by default) so Kermit sends it twice. With this command you can set the intercept character (ie the one to send twice) to another character.

TERMINAL *option*

Select one of the following terminal characteristics:

OFF sets emulation off, and its up to the attached terminal to respond to escape sequences sent from the remote host system.

DUMB Like off, but carriage return and line feed characters are the only control characters accepted. All other control characters are simply ignored. (Really a "Glass TTY").

EXTERNAL

Emulation is provided for by a routine in the system dependent part of Kermit. Attempting to set this option without having and externally supplied routine will returna "Not Implemented" error.

OFF All characters are passed directly to the terminal without any interpretation by Kermit.

VT52 When connected as a terminal to a foreign host, the micro emulates a VT52. VT52 emulation is set by default, except on micros that already have terminal functionality built in, such as the DEC VT180 and DECmate (these act as VT100-series terminals). Some systems emulate other terminals, like the ADM3A; see table 1-5.

QUIET Do not display any file transfer information onto the console. This mode is useful if you console takes a long time to update the display. Only the file name is displayed. DEBUGging information is not displayed even if selected.

REGULAR

Inverse of QUIET. All packets etc displayed, as ususal.

TIMER ON (or OFF)

Enable or disable the "fuzzy timer". The timer is off by default, because in the normal case KERMIT-80 is communicating with a mainframe KERMIT that has its own timer. Mainframe KERMIT timers tend to be more precise or adaptable to changing conditions. You should SET TIMER ON if you are communicating with a KERMIT that does not have a timer. You should SET TIMER OFF if you are communicating over a network with long delays.

USER *user-number*

Sets another user number to be active. Acceptable user numbers are 0 to 31, though it is recommended to use user numbers 0 to 15 only. This is really only useful for Winchester Systems with high disk capacities.

WARNING ON (or OFF)

Warn user of filename conflicts when receiving files from remote host, and attempt to generate a unique name by adding "&" characters to the given name. ON by default, which is equivalent to SET COLLISION RENAME.

SHOW Display all settable parameters. You will get a page or so of the status af all parameters that can be set using the SET command.

STATUS The same function as Show.

STAY Equivalent to SET NO-EXIT.

TAKE *filespec*

Take characters and commands from the specified file as if they were entered from the keyboard. This is useful if you want to set up a batch job. A command file can send, get, receive, set functions etc automatically. A TAKE command can be interrupted with ^C.

An automatic "TAKE KERMIT.INI" is executed from the default drive when Kermit-80 is loaded. This can be used to set defaults of band rate, parity, filetype, default drive etc.

If KERMIT.INI does not exist, control is given directly to the user.

TRANSMIT *filespec turnaround*

Send the specified file to the system on the other end of the connection as though it were being typed at the terminal, one line at a time. Each line sent is terminated with a carriage return, and any line feeds are stripped from the file sent. After each line has been sent Kermit waits for a character string from the host (eg a acrriage return). If not specified, a carriage return is assumed. No KERMIT protocol is involved. An asterisk (star) is sent to the console for every line sent, to indicate how the transfer is progressing. This is useful for sending files to systems that don’t have a KERMIT program. During transmission, you may type one of these single-character commands:

Control-C

Cease transmission, and drop into terminal emulation mode.

CR (carriage return) Re-transmit the previous line.

TYPE filespec

Type a local CP/M file or files on the CP/M screen. A Control-C will cancel the command and return to the Kermit prompt. A Ctrl-X will cancel the current file and go on to the next one, if any. Typing any other character while the file is being displayed will suspend the output. Another character will resume output.

VERSION

Show the name, edit number, and edit date of several of the modules that make up Kermit-80.

1.6. Kermit-80 Flavors

Many of the systems supported use an external terminal, rather than a built-in console. Kermit may be further customized for these systems by defining (at assembly time) the terminal type to be used. If the terminal type is unknown or does not match any of the existing terminal options, the generic "CRT" option may be selected. In this case, Kermit cannot do fancy screen control during file transfer; it simply types the file names, packet numbers, and messages in sequence across and down the screen. This works best if you can put your micro or terminal in "autowrap" mode; otherwise the packet numbers will pile up in the rightmost column; the filenames and messages will always appear on a new line, however. If no specific terminal has been selected, Kermit cannot do VT52 emulation; it can act as a "dumb terminal" (sometimes called a "glass TTY"), or else its own built in terminal firmware provides cursor control functions independent of the Kermit program.

1.6.1. Generic Kermit-80

"Generic Kermit-80" is an implementation of Kermit that should run on any 8080-compatible CP/M 2.2 system with no modification at all, or perhaps only a minor one. Unlike other Kermit-80 implementations, it contains no system-dependent manipulation of the serial port. All I/O is done with standard CP/M BIOS calls, and I/O redirection is done using the CP/M IOBYTE function, which, according to the Digital Research *CP/M Operating System Manual*, is an optional feature of any particular CP/M implementation. If your system does not provide the IOBYTE function, Generic Kermit-80 will not work; furthermore, not all systems that implement IOBYTE do so in the same way. The SET PORT command may be used to select the devices to be used for input and output. Table 1-1 lists the options to the SET PORT command and their effects.

`SET PORT` *`xxx`*	`input from`	`output to`
`CRT`	`CRT:`	`CRT:`
`PTR`	`PTR:`	`PTP:`
`TTY`	`TTY:`	`TTY:`
`UC1`	`UC1:`	`UC1:`
`UR1`	`UR1:`	`UP1:`
`UR2`	`UR2:`	`UP2:`

Table 1-1: Kermit-80 SET PORT Options

The default is SET PORT PTR. In all cases, the console (CON:) and list (LST:) devices used are those selected when Kermit is started.

The reason all Kermit-80 implementations aren't generic is that a good deal of speed is sacrificed by getting all services from the operating system. While a specific implementation of Kermit-80 may be able to operate at 4800, 9600, or even 56 Kilo baud, generic Kermit will fail to work on some systems at speeds in excess of 1200 baud. In addition, many features of Kermit require more specific knowledge of the hardware involved. Generic Kermit cannot send a BREAK signal, or change the baud rate, for example.

1.6.2. CP/M 3 Kermit

CP/M-3 Kermit (also known as CP/M-Plus Kermit) is a version of generic Kermit-80, and should run on most CP/M-3 (CP/M-Plus) systems. It uses the auxilliary port (AUX:) to communicate to the remote Kermit. The SET BAUD and SET PORT commands are not supported; nor can a BREAK be sent. Like generic Kermit-80, a terminal may be selected at assembly time.

1.6.3. System-Specific Versions

There are also many versions of Kermit-80 tailored to specific systems. Most of these operate uniformly, but some of them take advantage (or suffer limitations) of the specific system. Here are some of the special features for particular systems:

Amstrad: -- Two versions:

PCW 8256
: The PCW 8256/8512 with the serial inerafce attached.

CPC 6128
: The 664 with add on memory and 6128 are both supported. Both systems must run CP/M Plus, so the 664 will need an add on RAM pack and CP/M upgrade. A high speed transfer rate of 38k baud can be used between Amstrad computers.

ACCESS:
: Access Matrix computer using port J5. Supports SET BAUD-RATE for rates of 300-9600 baud.

Apple II -- four variations:

APMMDM:
: Apple with Z80 Softcard and Micromodem II in slot 2 Dialout capability provided in connect command; user is prompted for phone number if carrier is not present. During connect mode, `^]D` drops carrier. BYE command also causes carrier to be dropped.

AP6551:
: Apple with Z80 Softcard, and one of several 6551-based communication cards; the slot number is a compile-time parameter (default is slot 2). SET BAUD-RATE supported; speeds are 110-19200 baud.

APCPS:
: Apple with Z80 Softcard and CP Multi-Function Card. The slot number is again a compile-time parameter. SET BAUD-RATE is supported for baud rates from 50 baud to 19200 baud.

AP6850:
: Apple II with Z80 Softcard and a 6850-based USART in slot 2-the slot being a compile-time parameter. SET BAUD-RATE is not supported.

BBC:
: Acorn Computers BBC Computer with Acorn Z80 second processor running CP/M-80. Supports SET BAUD-RATE and can send breaks.

BigBoard II:
: Uses serial port A. To use port B, change mnport, mnprts, and baudrt and reassemble. Can generate BREAK. SET SPEED supported; speeds are 300-38400 baud.

Cifer:
: Originally coded for Cifer 1886 using the VL: port set as TTYI: and TTYO: but works successfully on 18xx and 28xx series machines.

There are now two versions, each with two variations: Either running CP/M Version 2.2 or 3.0, and either using the VL: or AUX: ports. The VL: port version can only use seven bits of data, so parity prefixing is required for binary file transfers. This restriction is removed by using the AUX: port. For those interested, the problem is due to the interprocessor link between the video and CPU (!) boards. The VL: port is on the video board, and the AUX: port on the CPU board, and the inter processor link can only transfer seven bits of data.

Supports SET SPEED, and can generate breaks on some models with a BREAK key.

Comart:
: Comart Communicator-Similar to Northstar equipment. Can generate BREAK.

Compupro:
: Based on Kermit 3.x, and has been merged into V4.09

CPT-85xx word processors:
: Can generate BREAK. SET SPEED supported; speeds are 50-9600 baud.

Cromemco:

Cromemco computers with TU-ART card. Supports SET BAUD-RATE (110-9600 baud).

DEC DECmate II word processor (with Z80 card):
Can generate BREAK.

DEC VT180 (Robin):
Three output ports, referred to as COMMUNICATIONS, GENERAL, and PRINTER. Can generate BREAK.

Digicomp Delphi 100:
SET SPEED supported; speeds are 50-19200 baud.

Discovery:
Action Computer Enterprises "Discovery" Multi-user Computer. Uses Port B on an 83U user board. Supports SET SPEED for 50-19200 baud. Can generate BREAK.

Epson:
Epson PX-8 with LCD display. Although it is quite different in displaying of Packet Information, it works as any other CP/M-80 Kermit. Supports SET SPEED and can generate BREAK.

Generic Kermit:
Two versions, one for CP/M version `2.2` and version 3. These systems use IOBYTE flipping (V`2.2`) and the AUX: device to communicate to the serial line. You may have to SET PORT xxx before the version `2.2` will work, as Kermit needs to know what device is the serial line.

Genie:
Eaca Video Genie.

Heath: Three Versions:

H8QUAD
for Heath-8 systems with the quad io board. This system has been derived from V3.x code. Note that this version will not run "as is" on H89 systems.

H89 For Heath-89 machines suports baud rates from 50 to 56,000 baud.

Z100
For Z-100 running CP/M-85. This version does not support setting of baud rates.

Intertec Superbrain: Two Versions:

BRAINA
For superbrain using AUX port. Breaks and SET BAUD both supported

BRAINM
As above, but using the MAIN port.

Ithaca:
Ithaca Intersystems based computer using the VIO card for all IO to the outside world. The system is strictly speaking a home-brew variant of the Ithaca machine, using an S100 cardcage without a front panel. It uses the Extended BIOS by EuroMicro of London. However, I see no reason for this version not running on a genuine Ithaca Intersystems machine. There are patches needed to the EuroMicro BIOS to make this version work.

Kaypro:
Should work on most Kaypro models, as well as some related systems (Ferguson BigBoard I, Xerox 820). For the newer Kaypros with multiple ports, Kermit uses the one labeled "serial data"; it cannot use the serial printer or internal modem ports (but it should be possible to modify the values for mnport, mnprts, and baudrt to do this). Can generate BREAK. SET SPEED supported; speeds are 50-19200 baud.

Lobo:
Lobo MAX-80. Supports SET SPEED and can generate BREAKS.

Merlin:
British Telecom Merlin M2215 (also Rair Black Box, possibly also the ICL PC?). Requires a terminal.

Microbee:
Microbee Systems computer made in Australia. Works on Microbee 56K (Series 2 APC), 64K (Computer in a Book), 128K (Dynamic), and 256K (256TC). Can generate BREAK. SET BAUD-RATE supported; speeds are 75-9600. All serial I/O is via software, not hardware. Simultaneous transmit and receive possible on all speeds except 75/1200, 1200/75, 4800, and 9600.

Micromate:
PMC 101 Micromate. Supports SET SPEED and can generate BREAK.

Micromint: Two versions

S6 The Ciarcia/Micromint sb-180 board with a 6Mhz procoessor. System requires a terminal.

S9 As above, but with a 9Mhz processor.

NCR:
Decisionmate 5. Uses the 2651 and is largely the same as the Ithaca Intersystems machine implementation.

Northstar: -- There are four versions available:

NORTHS:
Northstar Horizon with HS10-4 board. Supports SET SPEED and SET PORT.

HORIZON:
Northstar Horizon using the serial ports on the Mother board. Can generate BREAK.

BASICNS:
Basic Northstar Horizon using the printer port. Can generate BREAK.

ADVANT:
Northstar Advantage. Supports SET SPEED and can generate BREAK. Traps Control-0 in the system filter.

Morrow Decision I:
Uses the Multi-I/O board. Port 1 is the console, port 3 is the communications line. SET SPEED supported; speeds are 75-56000 baud.

Morrow Micro Decision I:

Nokia MicroMikko:
Will not echo control-O (which locks keyboard). SET SPEED supported; speeds are 75-9600 baud.

Ohio Scientific:
Doesn’t have screen control.

Osborne 1:
Uses serial line, not internal modem. Left-arrow key generates <DEL> ("delete" or "rubout" character) during connect mode. SET SPEED supported; speeds are 300 and 1200 baud. Now supports multi-sector buffering.

Research Machines: Two Versions:

RM380ZM:
380Z and 5.25" disks supports SET BAUD.RATE

RM380ZF:
380Z and 8" disks, otherwise as above.

Sanyo:
Sanyo MBC-1100. This version derived from Kermit V3.x

ScreenTyper:
Details unkown.

TRS-80: Three versions:

TRS80LB:
TRS-80 with Lifeboat CP/M

TRS80PT:
TRS-80 with Pickles and Trout CP/M

TRSM4:
TRS-80 Model 4 with Montezuma CP/M

Teletek:
Teletek Systemaster. Supports SET BAUD.

Telcon:
TELCON ZOBRA portable computer.

Torch:
Torch Unicorn 5 initially, but the code runs on other Z80 based CP/N (as in Nut!) systems. It uses the BBC Computer as a "Base processor", and is similar to the BBC version. The base processors RS423 port is used rather than any built in Modem. (UK telecoms legislation effectively makes modem control software tricky business...). Two potential versions exist-one using cursor positioning codes for a MCP and CCCP ROM combination of revision less than 1.00, the other version uses the additional facility MCP/CCCP versions greater than 1. Supports SET SPEED and can generate BREAKs.

Note that binary files must be transferred using SET PARITY to anything other than NONE! Parity is neither generated nor checked.

US Micro Sales:
S-100-8 based computer.

Vector Graphics:
Vector

Xerox:
Xerox 820.

Z80MU:
Development Kermit on a PC running the Z80MU Z80 and CP/M `2.2` development system. Allows development of the system independent modules to be done on an IBM PC or clone. Also allows the generation of new .HEX files, that may then be KERMITed to the target system. Note: Not all the BDOS or BIOS routines are supported, so avoid "unusual" BIOS/BDOS calls. (For example, DIR from within Kermit will fail as one of the BIOS routines returning disk parameters is not supported.)

1.7. Installation of Kermit-80

Kermit-80 was written originally for the Intertec SuperBrain in lowest-common-denominator 8080 code with the standard assembler, ASM (single source module, no macros, no advanced instructions), so that it could be assembled on any CP/M-80 system (the 8080 assembler is distributed as a standard part of CP/M-80, whereas the fancier Z80 or macro assemblers are normally commercial products). It has since been modified to run on many other systems as well. Kermit-80 should be able to run on any 8080-, 8085- or Z80-based microcomputer under CP/M with appropriate minor changes to reflect the port I/O and screen control for the system (see below).

The proliferation of new systems supported by Kermit-80 made the program grow so large and complicated that it had to be broken up into system-independent and system-dependent modules, as of version 4 (this was done by Charles Carvalho of ACC). Each module is composed of multiple files. This has reduced the time and disk space necessary for assembly; Kermit-80 may once again be assembled on a CP/M system with roughly 250Kbytes of space. The majority of the code does not need to be reassembled to support a new system. Unfortunately, it can no longer be assembled with ASM, since ASM does not support multiple input files. To allow it to be assembled on any CP/M system, the public-domain assembler LASM is included in the distribution kit. Kermit-80 may also be assembled with Microsoft's M80 (not supplied). In theory, any 8080 assembler supporting the INCLUDE directive ought to work, as well.

All versions of Kermit-80 are assembled from the same set of sources, with system dependencies taken care of by assembly-time conditionals within the system-dependent module (eventually, the system-dependent module will itself be broken up into multiple files, one for each system). The most important system dependencies are terminal emulation (when CONNECTed to the remote host) and screen handling, which are dependent on the individual micro's escape codes (these features are table driven and easily modified for other CP/M systems), and the lowest level I/O routines for the serial communications port. The port routines are best done only with BDOS calls, but some systems do not allow this, primarily because the BDOS routines strip the parity bit during port I/O, and the parity bit is used for data when transmitting binary files.

Kermit-80's I/O routines must check the port status and go elsewhere if no input is available; this allows for virtual terminal connection, keyboard interruption of stuck transmissions, etc. On systems that fully implement I/O redirection via the optional CP/M IOBYTE facility, this may be done by switching the IOBYTE definition. On others, however, IN/OUT instructions explicitly referencing the port device registers must be used.

CP/M-80 KERMIT versions 3.8 and later include a "fuzzy timer" that allows a timeout to occur after an interval ranging from 5 to 20 seconds (depending upon the speed of the processor and the operating system routines) during which expected input does not appear at the port. In this case, retransmission occurs automatically. In any case, you may type a carriage return during transmission to simulate a timeout when the transfer appears to be stuck.

1.7.1. Organization of Kermit-80

Kermit-80 consists of two modules, each of which is generated from multiple source files. The first module contains the system-independent code; the second module is configured for a particular system and merged with the system-independent module to produce a customized Kermit-80.

The distribution kit contains:

- the system-independent module, `CPSKER.HEX`;
- the system-dependent modules, `CPV*.HEX` (see table 1-2 and 1-3);
- the source files, `CPS*.ASM` and `CPX*.ASM`,
- the public-domain CP/M assembler, `LASM.*`,
- the public-domain CP/M load/patch utility, `MLOAD.*`

Symbol	Filename	System
`ACCESS`	`CPVACC`	Access Matrix
`ADVANT`	`CPVADV`	Northstar Advantage
`AP6551`	`CPVAPL`	Apple II, Z80 Softcard, 6551 ACIA in serial interface
`AP6850`	`CPVA65`	Apple II, Z80 Softcard, 6850 ACIA in Serial Iiterface
`APMMDM`	`CPVAPM`	Apple II, Z80 Softcard, Micromodem II in slot 2
`APCPS`	`CPVCPS`	Apple II, Z80 Softcard, with CPS multifunction card
`BASICNS`	`CPVBNS`	Northstar Horizon (terminal required)
`BBC`	`CPVBBC`	Acorn "BBC" computer with Acorn Z80 second processor
`BBII`	`CPVBB2`	BigBoard II (terminal required)
`BRAINM`	`CPVBRM`	Intertec Superbrain using the main port
`BRAINA`	`CPVBRA`	Intertec Superbrain using the Aux port
`CIFER2`	`CPVCIF`	Cifer 1886 using the VL: Serial port and CP/M V2.2
`CIFER3`	`CPVCI3`	Cifer 1886 using the VL: Serial port and CP/M V3.0
`CIFER2`	`CPVCA2`	Cifer 1886 using the AUX: Serial port and CP/M V2.2
`CIFER3`	`CPVCA3`	Cifer 1886 using the AUX: Serial port and CP/M V3.0
`CMEMCO`	`CPVCRO`	Cromemco with TU-ART card. Terminal required)
`COMART`	`CPVCOM`	Comart Communicator (terminal required)
`COMPRO`	`CPVPRO`	Compupro with Interfacer 4 (or 3). Terminal required.
`CPC`	`CPVCPC`	Amstrad CPC 664 and 6128 and CP/M 3
`CPM3`	`CPVCP3`	"Generic": CP/M 3.0 (CP/M Plus) systems (terminal req'd)
`CPT85XX`	`CPVCPT`	CPT-85xx wordprocessor with CP/M
`DELPHI`	`CPVDEL`	Digicomp Delphi 100 (terminal required)
`DISC`	`CPVDIS`	Action Computer Enterprises "Discovery" (terminal req'd)
`DMII`	`CPVDM2`	DECmate II with CP/M option
`GENER`	`CPVGEN`	"Generic": CPM 2.2 systems with IOBYTE (terminal req'd)
`GENIE`	`CPVGNI`	Video Genie
`H8QUAD`	`CPVH8Q`	Heath-8 with Quad 8 i/o board
`HEATH`	`CPVH89`	Heath/Zenith H89
`HORIZON`	`CPVHOR`	Northstar Horizon (terminal required)
`KPII`	`CPVKPR`	Kaypro-II (and 4; probably supports all Kaypro systems)
`LOBO`	`CPVLBO`	Lobo Max-80

"symbol" is the symbol used to select the target system, in `CPVTYP.ASM`;

"filename" is the name under which the module is supplied in the distribution.

Table 1-2: Systems supported by Kermit-80 (Part 1)

Symbol	Filename	System
M2215	CPVMRL	British Telecom Merlin/Rair Black Box (terminal required)
MBEE	CPVBEE	Microbee
MDI	CPVMDI	Morrow Decision I (terminal required)
MIKKO	CPVMIK	MikroMikko
MMATE	CPVMM	PMC 101 Micromate (terminal required)
MMDI	CPVUD	Morrow Micro Decision I (terminal required)
NCRDMV	CPVDMV	NCR Decision Mate V. (Terminal required?)
NORTHS	CPVNS	Northstar Horizon with HSIO-4 card (terminal req’d)
OSBRN1	CPVOSB	Osborne 1
OSI	CPVOSI	Ohio Scientific
PCI2651	CPVPCI	Ithaca Intersystems with VI0 card (terminal required)
PCW	CPVPCW	Amstrad PCW 8256/8512 with serial interface
PX8	CPVPX8	Epson PX-8
RM380ZM	CPVRMM	Research Machines 380Z with MDS (5.25" discs)
RM380ZF	CPVRMF	Research Machines 380Z with FDS (8" discs)
ROBIN	CPVROB	DEC VT180
S1008	CPVUSM	US Microsales S-100-8 (terminal required)
SANYO	CPVSAN	Sanyo MBC-1100
SB6	CPVSB6	Micromint SB-180 with 6Mhz CPU (terminal required)
SB9	CPVSB9	Micromint SB-180 with 9Mhz CPU (terminal required)
SCNTPR	CPVSCN	Screentyper
TELCON	CPVTEL	TELCON Zobra portable
TELETEK	CPVTET	Teletek Systemaster
TORCH	CPVTRC	Torch computers BBC-B with Z80 second processors
TRS80LB	CPVTLB	TRS-80 model II with Lifeboat 2.25C CP/M Display
TRS80PT	CPVTPT	TRS-80 model II with Pickles + Trout CP/M Display
TRSM4	CPVTM4	TRS-80 model IV
VECTOR	CPVVEC	Vector Graphics
XER820	CPVXER	Xerox 820
Z100	CPVZ00	Z-100 under CP/M-85
Z80MU	CPVZ80	Z80MU development system on a PC

"symbol" is the symbol used to select the target system, in `CPXTYP.ASM`;

"filename" is the name under which the module is supplied in the distribution.

Table 1-3: Systems supported by Kermit-80 (Part 2)

Symbol	Terminal type
`CRT`	Dumb terminal type. Does not do cursor addressing
`ADM3A`	Lear Seigler ADM 3A
`ADM22`	Lear Seigler ADM 22
`AM230`	Ampro 230
`H1500`	Hazeltine 1500
`SMRTVD`	Netronics Smartvid
`SOROQ`	Soroq IQ-120
`TVI912`	Televideo 912
`TVI925`	Televideo 925 or Freedom 100
`VT52`	Dec VT52 or equivalent (H19)
`VT100`	Dec VT100 or equivalent
`WYSE`	Wyse 100

"symbol" is the symbol used to select the target system, in `CPXTYP.ASM`;

"Terminal type" is the type of terminal "symbol" selects.

Table 1-4: Terminals supported by Kermit-80

1.7.2. Downloading Kermit-80

You'll need either a pre-configured `.COM` file or the system-independent module, `CPSKER`, in binary (`.COM`) or hex (`.HEX`) format and the system-dependent overlay for your system (from Tables 1-2 and 1-3). If your system is not listed in the table, get the generic CP/M `2.2` Kermit or the generic CP/M 3 Kermit. If you already have a version of Kermit on your micro and you want to install a new version, simply use your present version to get the new files. Transfer the files to your system and skip ahead to "merging the modules".

If you do not have a copy of Kermit on your micro, and you cannot borrow a Kermit floppy but you do have access to a mainframe computer with a copy of the Kermit-80 distribution, you should read this section.

There are several ways to get CP/M Kermit from a host system to your micro. The easiest is to "download" the necessary "hex" files into your micro's memory and then save them on the disk. If you have a terminal emulator program on your micro which can save a copy of the session to disk, connect to your host, and type the necessary files. Exit from the emulator, saving the session log, and edit the session log to extract the hex files. Skip ahead to "merging the files".

The following is a procedure which, though far from foolproof, should allow you to get a version of Kermit to your CP/M based micro. It depends upon the host prompt, or at least the first character of the host prompt, being some character that cannot appear in a hex file (the valid characters for hex files are the digits 0-9, the upper case letters A-F, the colon ‘‘`:`’’, carriage return, and line feed). As soon the prompt character is encountered, the transfer will terminate. If your host does not issue a prompt that will accommodate this scheme, you can achieve the same effect by adding an atsign ‘‘@’’ to the very end of the hex file before sending it from the host. The program below looks for an atsign (the normal DEC-20 prompt, hex 40). DECSYSTEM-10 users would look for a dot, hex 2E; VAX/VMS or UNIX users would look for a dollar sign, hex 24; UNIX C-Shell users would look for a percent sign, hex 26.

1. For CP/M `2.2` systems, connect to a floppy disk with plenty of free space. Run DDT and type in the following (the comments should not be typed in; they are there just to tell you what's happening): (Note that this wont work for CP/M Plus or `3.0` systems!)

```
-a100                   ;Begin assembling code at 100
     0100    LXI H,2FE       ;Where to store in memory
     0103    SHLD 200        ;Keep pointer there
     0106    MVI E,D         ;Get a CR
     0108    MVI C,4         ;Output to PUNCH (send to HOST)
     010A    CALL 5
     010D    MVI C,3         ;Input from READER (read from HOST)
     010F    CALL 5
     0112    ANI 7F          ;Strip parity bit
     0114    PUSH PSW        ;Save a and flags
     0115    MOV E,A         ;Move char to E for echo
     0116    MVI C,2         ;Output to screen
     0118    CALL 5
     011B    POP PSW         ;Restore A and flags
     011C    CPI 40          ;(or 4E,24,26,etc) System prompt?
     011E    JZ 127          ;Yes, have whole file in memory
     0121    CALL 17A        ;No, store another byte
     0124    JMP 10D         ;Read another byte
     0127    MVI A,1A        ;Get a Control-Z (CP/M EOF mark)
     0129    CALL 17A        ;Store it in memory
     012C    LXI H,300       ;Get memory pointer
     012F    SHLD 202        ;Store as DMA pointer
     0132    LDA 201         ;Get ’HI’ byte of memory pointer
     0135    STA 200         ;and store it as ’LO’ one
     0138    XRA A
     0139    STA 201         ;Zero ’HI’ byte (slow *256)
     013C    MVI C,16        ;Make NEW file
     013E    LXI D,5C        ;With FCB1
     0141    CALL 5
     0144    CALL 15E        ;Write 128 bytes (sector)
     0147    CALL 15E        ;Write another sector
     014A    LXI H,FFFF      ;Get a 16-bit Minus One
     014D    XCHG            ;into DE
     014E    LHLD 200        ;Get 256-byte counter
     0151    DAD D           ;Decrement
     0152    SHLD 200        ;and store back
     0155    MVI A,2         ;Check if
     0157    CMP L           ; 256-byte counter down to offset
     0158    JZ 183          ;Yes, we’re done
     015B    JMP 144         ;Keep writing..
     015E    LHLD 202        ;Get file-pointer
     0161    XCHG            ;into DE
     0162    MVI C,1A        ;Set DMA-address
     0164    CALL 5
     0167    MVI C,15        ;Write sector (128 bytes)
     0169    LXI D,5C        ;using FCB1
     016C    CALL 5
     016F    LHLD 202        ;Get file-pointer
     0172    LXI D,80        ;128-bytes
     0175    DAD D           ;added to file-pointer
     0176    SHLD 202        ;and save
     0179    RET             ;and return
     017A    LHLD 200        ;Get Memory-pointer
     017D    MOV M,A         ;Store character
     017E    INX H           ;Increment Pointer
     017F    SHLD 200        ;and save
     0182    RET             ;and return
     0183    MVI C,10        ;CLOSE file
     0185    LXI D,5C        ;using FCB1
     0188    CALL 5
     018B    JMP 0           ;Force WARM BOOT
```

```
0179
-^C                        ;(Type Control-C) Return to CP/M
A>SAVE 1 FETCH.COM         ;Save program, we need to run it twice.
```

Figure 1-1: Bootstrap program for Kermit-80 and CP/M Version 2.2

Alternatively, an assembler source file for this program is distributed with CP/M Kermit as `CPKFET.ASM`. You might prefer to type the assembler version in and assemble and load it (ASM CPKFET, LOAD CPKFET, or MASM CPKFET, MLOAD CPKFET), to let the assembler and loader catch any typing errors.

2. Connect to your host using a terminal or a terminal emulation program. Ensure that your host does not have your terminal in "page mode" (does not pause at the end of each screenful).

3. Tell the host to display the first hex file (the system-independent module) at your terminal, e.g. give a command like `TYPE CPSKER.HEX`, *without a terminating carriage return*.

4. Return to your micro by switching the cable from the terminal to the micro, or by terminating the micro's terminal program.

5. Make sure your IOBYTE is set so that `RDR:` and `PUN:` correspond to the I/O port that is connected to the host (this would normally be the case unless you have done something special to change things).

6. Load the program you entered in the first step with DDT, and use it to capture the first hex file:

```
DDT FETCH.COM
-icpsker.hex              ;Setup FCB for file CPSKER.HEX
-g100,179                 ;Execute the program.
```

   Now there should be a file `CPSKER.HEX` on your connected disk.

7. Return to the host, and tell it to display the second hex file (the system-dependent module for your configuration). Again, do not type the terminating carriage return.

8. Return to your micro, and run the capture program again:

```
DDT FETCH.COM
-icpxovl.hex              ;Setup FCB to create CPXOVL.HEX
-g100,179                 ;Execute the program.
```

   Now there should be a file `CPXOVL.HEX` on your connected disk. Replace `CPXOVL.HEX` in this example with the appropriate overlay file for your system.

Merging the files:

1. For purposes of illustration, we will assume the system-dependent overlay is called "cpxovl.hex". The two hex files may be combined with MLOAD or DDT. If you already have a running Kermit, you can transfer `MLOAD.HEX` to your system and create `MLOAD.COM` by running LOAD. If you're bootstrapping Kermit, you could transfer `MLOAD.HEX` to your system the same way you got the other two `.HEX` files, but it's probably simpler to use DDT to get Kermit running, and get MLOAD later if you need it.

2. Using MLOAD, the two pieces may be easily merged:

```
A>mload kerm411=cpsker,cpxovl
(Some messages about program size, etc...)
A>
```

3. If you don't have MLOAD running, it's a bit more complex:

```
A>ddt cpsker.hex
NEXT  PC
3500 0100
-icpxovl.hex
-r
NEXT  PC
xxxx 0000
-^C
A>save dd kerm411.com
```

The page count ("dd") used in the SAVE command is calculated from the last address ("xxxx") given by DDT in response to the R command: drop the last two digits and add 1 if they were not zero, then convert from hexadecimal (base 16) to decimal (base 10): 684F becomes 69 hex, which is 105 decimal (5 times 16 plus 9) -- but 6700 becomes 67 hex, or 103 decimal (consult an introductory computing book if you don't understand number base conversion).

4. If you are using the Z80MU CP/M and Z80 development toolkit on an IBM PC or clone, then follow the same instructions as for a genuine CP/M system. When you have loaded your file, you will have to ship the .COM or two .HEX files to the target CP/M system. (Possibly using a previous issue of Kermit?)

5. Note that CP/M hex files have checksums on each line. If there were any transmission errors during the downloading process, MLOAD or DDT will notice a bad checksum and will report an error (something like "Illegal Format"). If you get any errors during loading, either fix the hex file locally with an editor, or repeat the transfer.

You now should have a running version of Kermit-80, called `KERM411.COM`.

Test your new Kermit by running it. If it gives you a prompt, it might be OK. (don't delete your old one yet...). Instead of a prompt, you could get one of two messages indicating that the configuration information is invalid:

```
?Kermit has not been configured for a target system
```

or

```
?Consistency check on configuration failed
```

Of course, neither of these messages should appear if you're building Kermit from the distribution kit. The first message indicates that the overlay was not found where the system-independent module expected to find it, probably because the overlay address is incorrect; the second indicates that the version of CPXLNK used in the system-dependent module is incompatible with the system-independent module.

Once you are satisfied that KERMIT40 works correctly, you should rename your old `KERMIT.COM` to something else, like `OKERMIT.COM`, and rename `KERMIT40.COM` to `KERMIT.COM`.

1.7.3. Assembling Kermit-80 from the sources

Kermit-80 is built in two pieces from the following files:

The system-independent files:

`CPSKER.ASM`	header file
`CPSDEF.ASM`	definitions for both KERMIT and KERSYS
`CPSMIT.ASM`	initialization, main loop, miscellaneous commands (BYE, EXIT, LOG, SET, SHOW, STATUS, and VERSION)
`CPSCOM.ASM`	second part of commands, status and set file
`CPSPK1.ASM`	part 1 of the KERMIT protocol handler (SEND, RECEIVE, LOGOUT,and FINISH commands)
`CPSPK2.ASM`	part 2 of the KERMIT protocol handler
`CPSREM.ASM`	REMOTE routines (FINISH, BYE and LOGOUT in CPXPK*.ASM)
`CPSSER.ASM`	SERVER routines (for the future)
`CPSTT.ASM`	the transparent commands (TRANSMIT, CONNECT)

`CPSCPM.ASM`	CP/M commands (DIR, ERA, USER, TYPE, PRINT, COPY)
`CPSWLD.ASM`	the wildcard handler
`CPSCMD.ASM`	the command parser
`CPSUTL.ASM`	utility routines and data
`CPSDAT.ASM`	data space and the overlay definitions
`CPXLNK.ASM`	linkage area description

The system-dependent files:

`CPXTYP.ASM`	system selection
`CPXLNK.ASM`	system overlay specification and jump table
`CPXCOM.ASM`	common routines for all systems
`CPXSWT.ASM`	system selector or switcher

One of:

`CPXSYS.ASM`	family file for some system-specific code
`CPXTOR.ASM`	family file for Torch, Superbrain, PCI2651 etc
`CPXNOR.ASM`	family file for Northstar and Comart machines
`CPXMRL.ASM`	family file for British Telecom Merlin/Rair Black Box
`CPXSB.ASM`	family file for Micromint SB-180 systems
`CPXCIF.ASM`	family file for Cifer systems
`CPXHEA.ASM`	family file for Heath/Zenith systems
`CPXAPP.ASM`	family file for Apple II systems
`CPXPCW.ASM`	family file for Amstrad PCW 8256/8512 machines
`CPXBBI.ASM`	family file for BigBoard, Kaypro and Xerox 820 systems
`CPXBEE.ASM`	Microbee
`CPXSYO.ASM`	family file for Sanyo MBS-1100 systems
`CPXTM4.ASM`	family file for Tandy Model 4 with CP/M systems
`CPXGNI.ASM`	family file for Video Genie systems
`CPXPRO.ASM`	family file for Compupro systems
`CPXZ80.ASM`	family file for the Z80MU development system

and if you use a terminal,

`CPXVDU.ASM`	display codes for VDUs etc. Not always required

The system-independent module contains all of the system-independent files except for `CPXLNK.ASM`, which is assembled into the system-dependent module to provide the structures needed to connect the two modules. As distributed, the system-independent module is named `CPSKER.HEX`. If you have a copy of `CPSKER.HEX`, you do not need to reassemble the system-independent module to configure Kermit for your system.

The system-dependent module consists of `CPXTYP.ASM`, `CPSDEF.ASM`, `CPXLNK.ASM`, `CPXSWT.ASM`, `CPSCOM.ASM`, one of the family files `CPXSYS.ASM`, `CPXTOR.ASM`, `CPXMRL.ASM`, `CPXSB.ASM`, `CPXCIF.ASM`, `CPXHEA.ASM`, `CPXBBI.ASM`, `CPXTM4.ASM`, `CPXGNI.ASM`, `CPXNOR.ASM`, `CPXAPP.ASM`, `CPXPCW.ASM`, or `CPXPRO.ASM`, and possibly `CPXVDU.ASM`, if your system uses a terminal for the console. One copy of the system-dependent module is supplied already assembled for each supported system; the filename may be obtained from tables 1-2 and 1-3. If a terminal is required for a system, a CRT (glass TTY device) has been selected.

After assembling the two pieces separately, they are combined with DDT or MLOAD into a system-specific Kermit.

If you want to rebuild the system-independent module, the only change you may need to make is to select the assembler to be used, in `CPSKER.ASM`. Define one of MAC80, M80, or LASM to TRUE to select it as the assembler; the others should be defined FALSE.

Assuming you have the Microsoft Macro Assembler package (M80/L80), you'll need to do the following:

```
A>m80 cpsker=cpsker.asm
A>l80 /p:100,cpsker,cpsker/n/e
```

This will produce `CPSKER.COM`.

If you are using LASM instead, do this:

```
A>lasm cpsker
```

LASM will generate `CPSKER.HEX` and `CPSKER.PRN`. LASM allows options to be specified in the same way as the standard assembler, ASM, so the command

```
A>lasm cpsker.abz
```

will read the source files from drive A, send the `.HEX` file to drive B, and suppress the listing file.

If you are using the Z80MU development system on an IBM PC or clone, then assemble your files using either LASM and MLOAD or M80 and L80, as if you were using a genuine CP/M-80 system. Note that you will still have the problem of transferring your assembled files to the target CP/M system.

If you want to generate a system-dependent overlay for a particular system, or want to change the terminal supported, you'll need to check three areas in `CPXTYP.ASM`:

First, the overlay start ADDRESS. The symbol "ovladr" is EQUated to the address of "LNKFLG" in the system-independent module, as the starting address of the overlay (7000H for version `4.11`). You'll need to know this value if you're building the overlay with M80/L80. You won't normally need to change this value.

Second, the assembler being used. Again, define one of MAC80, M80, and LASM to be TRUE to select it, and define the others to be FALSE. The two modules (system-independent and system-dependent) do not need to be built with the same assembler.

Third, the system configuration. Locate your system in tables 1-2 and 1-3, then define the appropriate symbol TRUE, and the rest FALSE. If the system comes with a builtin console terminal, define all the terminal switches FALSE. If the system uses an external terminal as the console, locate the terminal in table 1-5 and define the appropriate symbol TRUE, and the remainder FALSE. If the terminal is not listed in table 1-5, use the CRT switch; in this case, VT52 emulation is not supported.

In addition, there are a few general and system-specific symbols which may be altered to fit your system:

APSLOT	For Apple with 6551 ACIA, defines the slot number of the serial card
CPUSPD	Processor speed in units of 100KHz (currently used only for bbII and kpII for timing loops)
TAC	For users connecting through ARPAnet TACs: set to TRUE if you wish the default TACTRAP status to be ON. (This may be overridden with the SET TACTRAP command). If you're not connecting through a TAC, set tac to FALSE and ignore tacval.
TACVAL	For ARPANET TAC users: defines the default TAC intercept character (may be overridden with the SET TACTRAP command).

If you are just assembling an existing configuration, you'll need to edit `CPXTYP.ASM` only. If you are adding support for a new system, you should not modify `CPSDEF.ASM` or `CPXLNK.ASM`; if you do, you'll have to change the system-independent module also. Eventually, `CPXSYS.ASM` will be split into separate files, each of which will generate one or more related systems. When this happens, you'll want to pick the one closest to your system to use as a starting point.

After editing `CPXTYP.ASM` as necessary, assemble and link the overlay as follows:

- With M80 (where "xxxx" is the hex value of ovladr from `CPXLNK.ASM`):

  ```
  A>m80 cpxtyp=cpxtyp.asm
  A>l80 /p:xxxx,cpxtyp,cpxtyp/n/x/e
  ```

- With LASM:

```
A>lasm cpxtyp
```

With an IBM PC or clone using the Z80MU softwrae, follow the instructions as if you were using a real CP/M system.

The overlay (`CPXTYP.HEX`) may then be merged with the system-independent module as described above (creating a runnable Kermit from the distribution kit).

If you are using the Z80MU development system on a PC, and already have a running Kermit-80 v3.9 or later, you can merge the two `.HEX` files into a `.COM` file with LINK80 (TOPS 10/20), MLOAD (Z80MU), L80 (Z80MU), and transfer the new `.COM` file to your micro with Kermit:

- Z80MU on a PC and MLOAD:

```
@MLOAD KERNEW=CPSKER,CPXTYP
```

- Z80MU on a PC and C80:

```
@L80 /P:xxxx,CPXTYP,CPXTYP/N/X/E
```

producing `KERNEW.COM`.

Symbol	Terminal description
crt	Basic CRT, no cursor positioning
adm3a	ADM3A Display or lookalike
adm22	ADM22 Display or lookalike
am230	Ampro 230
h1500	Hazeltine 1500
smrtvd	Netronics Smartvid-80
soroq	Soroq IQ-120
tvi912	TVI 912
tvi925	TVI 925, Freedom 100
vt52	VT 52 or VT52 emulator such as Heath H19, H29, etc.
vt100	VT 100 or emulator (most ANSI terminals should work)
wyse	Wyse 100

Table 1-5: Terminals known to Kermit-80

1.8. Adding Support For A New System

Kermit-80 is built from a common set of source files; the system-dependent module makes heavy use of conditional assembly (this complication will be removed in future releases). The system dependencies arise from attempts to answer some questions:

1. *What kind of terminal is to be supported?*

   For many micros, the console is an integral part of the system, but others can use an external terminal. In either case, the commands to manipulate the screen (position the cursor, erase the screen, etc) must be defined.

2. *How is the serial line accessed?*

   For systems supporting the IOBYTE function, this is straightforward; the symbol "IOBYT" is defined TRUE. If the serial line is accessed with IN and OUT instructions, it may be possible to use the simple I/O routines provided. In this case, the symbol "INOUT" is defined TRUE, the MNPORT and MNPRTS are defined to be the data and control addresses, respectively, and bit masks for testing for "input data available" and "output buffer empty" must be defined. If the interface is strange, leave IOBYT and INOUT set to FALSE, and provide the I/O routines.

3. *What initialization is necessary?*

   You may wish to set the baud rate or configure the serial line at startup. Examples for a number of devices are present.

4. *What special features are to be supported?*

   You may want to provide the capability to select one of several serial lines with the SET PORT command, or to change the speed of the serial line with the SET SPEED command. To do this, you'll need to build a command table, using the systems already supported as examples. The ability to send a BREAK signal is desirable. Again, examples for several different interfaces (ACIA, SIO, etc) are present.

5. *Do you want to design an external terminal type?*

   There is a jump entry in the overlay file to allow users to add their own termainl emulator. If you write the code for such an emulator, you must load this jump address with the address of your emulator, and SET TERMINAL EXTERNAL from within Kermit. All characters will be passed to this routine during connect mode.

1.9. Notes on New Features in Kermit-80 Version 4

- *Debugging aids:* SET DEBUG ON will add two fields to the SEND/RECEIVE display, labelled "Spack" and "Rpack". These display the last packet sent and received. Of course, this slows down the transfer, especially if the console is an external terminal. SET DEBUG OFF removes these fields. The VERSION command displays the name, edit number, and edit date of several of the modules that make up Kermit.

- *TAC support:* ARPAnet TACs (and many other communication devices such as terminal concentrators, modems, port contention units, network PADs, etc) use a printing character (like "@") as an intercept character, to allow commands to be issued to the TAC, or modem, etc. In order to send this character to the host, it must be typed twice. The command "SET TAC CHARACTER" to Kermit enables the "TACtrap" and asks the user to specify the TAC intercept character. This character will be automatically doubled when it appears in Kermit protocol messages (sent by the SEND or RECEIVE commands) or when it appears in a file being sent with the TRANSMIT command. It is not automatically doubled when typed by the user in CONNECT mode. "SET TAC ON" enables the TACtrap but does not change the TAC intercept character, which is initially "@". "SET TAC OFF" disables the TACtrap.

- *File buffering:* Previous versions of Kermit-80 buffered only one sector (128 bytes) at a time during file transfer operations. This version buffers 16Kbytes at a time, reducing the number of times the floppy drive must be spun up and down, and increasing the effective throughput of the link. If the disk transfer rate is too slow, however, the remote Kermit may time out and retransmit packets. This will show up on the screen in the "Retries:" field; if this occurs after disk activity, you may want to increase the timeout value on the remote Kermit, SET BUFFER <new value> while in Kermit,or reassemble Kermit with a smaller value for MAXSEC (in `CPSDEF.ASM`) This buffer is also used by the TRANSMIT command; the log file enabled by the LOG command is still written a sector at a time.

This section is intended for people wanting to implement their own versions of Kermit-80 for computers not already defined.

The system independent code communicates to routines for a specific system through a set of tables. These tables are defined in CPXLNK.ASM, and should not be modified between revisions of Kermit. If an entry is added or deleted, then the whole of Kermit-80 needs reassembling. Make sure that the changes to CPXLNK.ASM are duplicated in CPSUTL.ASM, which has the system independent equivalent of CPXLNK.ASM.

The following entries/definitions apply to revision 4.09. There have been three additional entries since revision 4.05.

The table is split into three sectors; The first section defines two byte "words" giving 16 bits of interface data; The second set is a set of jumps to various functions, and finally the third set a set of pure data bytes.

1.9.1. Interface Data.

LNKFLG	Must be first entry in overlay at overlay address. Is a two byte address giving the size of the linkage table. This is used to check for consistency of overlay's
ENTSIZE	Length of entry table, also used for consistency checking after the overlay. Currently 6
SYSEDT	The address of a dollar-terminated string giving the overlay revision level and date. Points to a string like: CPXSYS.ASM(33) 4-JUN-1986$
FAMILY	The address of a dollar-terminated string giving the Family overlay revision level and date. If the system is in CPXSYS.ASM rather than a particular Family overlay, it is simply a pointer to $

1.9.2. Jump Table.

This is split into three main sectors-

1. Input/Output routines

2. Screen formatting routines

3. other system dependent routines

SELMDM

Parameters	None
Returns	None
Description	selects the modem port. Most systems do nothing and simply return. HL,DE and BC registers preserved.

OUTMDM

Parameters	None
Returns	None
Description	Output the character in E register to the communications line. BC,DE,HL registers preserved.

INPMDM

Parameters	None
Returns	Accumulator either 0 or character from comms line if available
Description	Check modem for character and if so, return it in A. HL,DE,BC registers preserved, flags and accumulator lost.

FLSMDM

Parameters	None
Returns	None
Description	Clear any pending characters in the input buffer from the modem. No registers are preserved.

SELCON

Parameters	None
Returns	None
Description	Select the console. This is a null subroutine for most systems, but for IOBYTE systems selects the console.

OUTCON

Parameters	Character in E
Returns	None
Description	Send the character in E to the console. Any quirks of system responding in an odd manner should be handled. No registers preserved.

INPCON

Parameters	None
Returns	Zero or character in A.
Description	Get a character from the console or return a null if no character to be read. No registers are preserved.

OUTLPT

Parameters	Character in E
Returns	None
Description	Send the character in E to the printer. The console is selected. Only DE registers are preserved

LPTSTAT

Parameters	None
Returns	00H or 0FFH in A register
Description	Test the printer to see if it is ready to receive a character to be printed. If a 00H is returned then the printer is ready to receive a character.

EXTTER

Parameters	Character to be sent to the user supplied terminal emulator in the E register
Returns	None
Description	If the user has supplied a terminal emulator in the overlay code, EXTTER will be a JMP <non zero address>. If SET TERMINAL EXTERNAL has been set, all caharcters will be passed verbatim to this terminal emulator. If there is no external emulator, this code will never be called. The user should reset terminal conditions on initialisation of both the system and before CONNECT. All registers should be preserved.

XBDOS

Parameters	Any required for calling BDOS
Returns	Any expected from the called BDOS routine
Description	This is an alternative entry to BDOS. This entry will also check the printer status etc. For full details see the code for the BDOS trap in CPSUTL.ASM.

2b)

CLRLIN

Parameters	None
Returns	None
Description	Clear the current line on the terminal

CLRSPC

Parameters	None

Returns	None
Description	Erase the current position (after a backspace)

DELCHR

Parameters	None
Returns	None
Description	Make delete (7FH) look like a backspace. Some systems do a backspace, space, backspace automatically others have to simulate it

CLRTOP

Parameters	None
Returns	None
Description	Clear the screen and place the cursor at the top LH corner

SCREND

Parameters	None
Returns	None
Description	Place the cursor on the line for the Kermit-80 prompt after a file transfer. (Usually line 13)

SCRERR

Parameters	None
Returns	None
Description	Move cursor to the error message field on the file transfer format screen

SCRFLN

Parameters	None
Returns	None
Description	Move the cursor to the filename field

SCRNP

Parameters	None
Returns	None
Description	Move the cursor to the packet count field

SCRNRT

Parameters	None
Returns	None
Description	Move cursor to the retry count field

SCRST

Parameters	None
Returns	None
Description	Move cursor to the status field

RPPOS

Parameters	None
Returns	None
Description	Move to the receive packet field (debugging use)

SPPOS

Parameters	None
Returns	None
Description	Move to the send packet field (for debugging use)

2c)

SYSINIT

Parameters	None
Returns	None
Description	Initialize the system specific items. No registers are preserved. Any initialization is done once only when Kermit-80 is first loaded.

SYSEXIT

Parameters	None
Returns	None
Description	Program termination. De-initialize anything in preparation for a return to CP/M

SYSCON

Parameters	None
Returns	None
Description	Initialize anything before entering the connect state.

SYSCLS

Parameters	None
Returns	None
Description	System dependent close routine when exiting connect state

SYSINH

Parameters	None
Returns	None
Description	Help routine to test for any extensions to the escape menu during the connect state. If a system has any special feature it can use during connect mode, then it can be tested as <escape-character>xxx. This entry is a string for printing to the console for an <escape-character>? Often used for generating breaks or controlling a modem.

SYSINT

Parameters	None
Returns	None
Description	This is a test-and-jump on receipt of an escape sequence not understood by Kermit-80. If the character in A is not recognized by your version of Kermit=80, do a rskip

SYSFLT

Parameters	Character in E
Returns	Character in E. Either a 00H or anything else in A
Description	Test the character in E. If it may not be printed to the console, set A to zero. All other registers preserved. NB <XON>,<XOFF>,<DEL>,<NULL> are always rejected.

SYSBYE

Parameters	None
Returns	None
Description	System dependent processing for the BYE command. (eg hang up the phone)

SYSSPD

Parameters	Value from table in DE
Returns	None
Description	The system dependent code for baud rate change. DE contains the two byte value from the baud rate table. This value is also stored in "SPEED"

SYSPRT

Parameters	Value in DE
Returns	None
Description	The system dependent code for setting the port. The parameters are passed in DE, which are obtained from the port tables

SYSSCR

Parameters	String pointer in DE
Returns	None
Description	Setup the screen display for file transfer. The Kermit version string is pointed to by DE. If the terminal is not capable of cursor addressing (eg dumb glass TTY) then only the screen is cleared and the version string is printed.

CSRPOS

Parameters	Row number in B, column number in C
Returns	None
Description	Move the cursor to row B, column C where B=1,C=1 is top LH corner of screen. The routine should first end a "cursor position" leading string (up to four characters) then use the parameters given to complete the versions cursor position function

SYSSPC

Parameters	None
Returns	K bytes free in HL
Description	Get the amount of free disk space on the selected disk drive. This could be in the system independent code. Automatically detects CP/M V2.2 or V3.0. No registers saved.

MOVER

Parameters	Source Pointer in HL Destination Pointer in DE Byte count in BC
Returns	None
Description	Move (BC) bytes from (HL) to (DE) Z80 based systems do an LDIR, while 8080 systems do it as a loop. All registers destroyed

PRTSTR

Parameters	$ terminated string pointed to by DE
Returns	None
Description	Print the string onto the console.

3)

PTTAB	WORD	Points to VT52 equivalent escape sequences.
SPDTAB	WORD	Address of baud-rate command table, or 0 if table does not exist
SPDHLP	WORD	Address of baud-rate help table, or 0 if SET SPEED is not supported.
PRTTAB	WORD	Address of port command table or 0 if SET PORT is not supported.
PRTHLP	WORD	Address of port help table or 0 if SET PORT is not supported
TIMOUT	BYTE	FUZZY-TIMER. Set to value suitable to your system (depends largely on CPU speed)
VTFLG	BYTE	VT52 emulation flag. Set to 0 if terminal emulates a VT52, 01 if emulation is required, or 0FFH if emulations not possible (eg for "CRT")
ESCCHR	BYTE	default escape character-usually control-] but sometimes control-\
SPEED	WORD	Storage space for baud-rate. Set to 0FFFFH as baud rates are initially

unknown. Note that the STATUS routine only looks at the first (least significant) byte.

PORT	WORD	Storage space for port. Set to 0FFFFH as ports may not be implemented, and is initially unknown
PRNFLG	BYTE	Printer copy flag-if O no copy. Anything else => copy to printer
DBGFLG	BYTE	Debugging flag. If O then no debugging to be done. (ie writing of debugging info during a file transfer)
ECOFLG	BYTE	Local ECHO flag (default is off)
FLWFLG	BYTE	File warning flag. If set to 1 will not overwrite files already existing on disk with some-named files being transferred
IBMFLG	BYTE	IBM system is the host-assume IBM file transfers etc
CPMFLG	BYTE	Flag indicating type of CP/M files to be transferred. Default setting - DEFAULT
PARITY	BYTE	Type of parity in use 0 = Even parity 3 = Mark parity 6 = No parity (8th bit is data) 9 = Odd parity 12 = Space parity
SPSIZ	BYTE	Size of send packet
RPSIZ	BYTE	Size of receive packet
STIME	BYTE	Send timer (time-out)
RTIME	BYTE	Receive timer (time-out)
SPAD	BYTE	Send Padding (default=0)
RPAD	BYTE	Receive Padding (default=0)
SPADCH	BYTE	Send Padding character (default=NULL)
RPADCH	BYTE	Receive Padding character (default=NULC)
SEOL	BYTE	Send EOL character (default=CR)
REOL	BYTE	Receive EOL character (default=CR)
SQUOTE	BYTE	Send quote character (default=#)
RQUOTE	BYTE	Receive quote character (default=#)
CHKTYP	BYTE	Ascii value of checktype 31H="1"=checktype1 (6bits) 32H="2"=checktype2 (12bits)

		33H="3"=CCITT checksum (CRC) Default is 31H("1")
TACFLG	BYTE	If set to on (non zero) send the TACCHR twice. This is for ARPA TAC users, where the TAC swallows one "wakeup" character. If sent twice the TAC will pas one on and go back to normal mode.
TACCHR	BYTE	Desired TAC character. It is ignored if TAC trapping is turned off. Value to send twice if TAC interception is set on. Default=0, but set to commercial AT if the conditional assembly flag TAC is set true
BUFADR	WORD	Address of Multi-Sector buffering for I/O
BUFSEC	BYTE	The number of bytes the big buffers can hold. Default is 1. (0=256 sectors).
FFUSSY	BYTE	Indicates if funny characters may be used in CP/M file names (eg `<>.,;?#[]`) If zero, allow anything. Default is nonzero.
BMAX	SPACE:(2bytes)	Highest block number on selected disk drive
BMASK	SPACE:(1byte)	(Records/block)-1
BSHIFTF	SPACE:(1byte)	Number of shifts to multiply by rec.block
NNAMS	SPACE:(1byte)	Counter for file-names per line

1.10. Future Work

Work that needs to be done in future releases includes:

- Merge in support for additional CP/M-80 systems, particularly those for which support was recently added to the monolithic v3.x source.

- Break up `CPXSYS` into discrete source files, one for each system. These source files should serve as simple models for adding support for new systems to Kermit-80 -- only the very basic screen definitions, flags, i/o primitives, initializations, and so forth should appear in each system-dependent file.

- Addition of missing features -- compression of repeated characters during packet transmission, transmission of file attributes (particularly size, so that "percent done" can be displayed for both incoming and outbound files), command macros, more advanced login scripts, remote operation and server mode, etc etc. Any offers??

Index

Table of Contents

List of Figures

List of Tables