10. PC-DOS and MS-DOS KERMIT-86

Program: Daphne Tzoar, Columbia University, with contributions from Jeff Damens (Columbia), Dave King (CMU), Herm Fischer (Litton Data Systems), and others.
Documentation:
Frank da Cruz, Columbia University; Herm Fischer, Litton Data Systems (Van Nuys CA)
Version: 2.25
Date: February 1984

KERMIT-86 is a program that implements the KERMIT file transfer protocol for the IBM PC and several other machines using the same processor family (Intel 8088 or 8086) and operating system family under PC-DOS or MS-DOS (henceforth referred to collectively as MS-DOS), versions 1.1, 2.0, and 2.1. This section will describe the things you should know about the MS-DOS file system in order to make effective use of KERMIT, and then it will describe the KERMIT-86 program.

MS-DOS KERMIT runs on a variety of systems, including the IBM PC and XT, the the Heath/Zenith 100, HP-150, the Seequa Chameleon, the Victor 9000, the Tandy 2000, the Compaq Portable, the Columbia MPC, and others. This document concentrates on the IBM PC/XT implementation; the others will be (possibly complete) subsets of that (see Section 10.5 for details about support for other MS DOS systems).

10.1. 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 distinction between pre-MS-DOS 2.0 file names and newer file names which allow directory paths.

MS-DOS FILE SPECIFICATIONS

MS-DOS file specifications are of the form

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

where the DEVICE is a single character identifier (e.g., A for the first floppy disk, C for the first fixed disk, D for a RAM disk emulator), PATHNAME is up to 63 characters of identifier(s) (up to 8 characters each) surrounded by reverse slashes (or ".." for parent or "." for current directory), 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. Pathname is normally omitted, and cannot be specified for MS-DOS 1.x or with those commands which allow MS-DOS 1.x use (e.g. pathnames can only be accepted by commands which are specific to MS-DOS 2.x). Device and directory pathnames, when omitted, default to the user's current (or "defaulted") disk and directory path (path="."). Thus NAME.TYPE is normally sufficient to specify a file, and only this information is sent along by KERMIT-86 with an outgoing file.

The device, path, name, and type fields may contain uppercase letters, digits, and the special characters "-" (dash), "_" (underscore), and "$" (dollar sign). (For use only among MS-DOS processors, additional filename special characters allowed are "#&!%(){}'`". DOS 1.x allows others as well.). There are no imbedded or trailing spaces. Other characters may be not be included within the MS-DOS environment (e.g. quoted characters are not permissible). The fields of the file specification are set off from one another by the punctuation indicated above.

The device field specifies a physical or "logical" device upon which the file is resident. The directory pathname identifies an area on the device, for instance the area belonging to the logical ownership of the file. KERMIT-86 does not transmit the device or pathname directory fields to the target system, and does not attempt to honor device or directory fields that may appear in incoming file names.

The name field is the primary identifier for the file. The type, also called the "extension", 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 an executable program produced by linking FOO.OBJ, and so forth.

The 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 (all BASIC source files) in the current directory.

FOO.* Files of all types with name FOO.

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

F?X*.* All files whose names start with F and contain X in the third position, followed by zero or more characters.

?.* All files whose names are exactly one character long.

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.

KERMIT-86 uses the ? character for help while commands are being typed, so the single-character wildcard in KERMIT commands is = rather than ?, for example

```
Kermit-86>send =.*
```

The KERMIT-86 user must bear in mind that other (non-MS-DOS) systems use different wildcard characters; for instance KERMIT-20 uses % instead of the ? as the single character wildcard. When using KERMIT-86 to request a wildcard file group from a KERMIT-20 server, the Kermit-86 "=" must be replaced by DEC-20 "%" characters.

TEXT FILES AND BINARY FILES

The MS-DOS systems store files as bulk collections of 8 bit bytes, with no peculiar differences between text, program code, and binary files. Since a non-MS-DOS receiving system might need to know file type distinctions, the user might need to use various SET functions on the remote system to inform it that the incoming file is of some particular (non-default) type. In transmitting files between KERMIT-86'es, regardless of file contents, the receiving MS-DOS system is equally capable of processing text, code, and data (and is, in fact, not knowledgable of the usage of the bytes in the file).

ASCII files are presumed to have recognizable characteristics (carriage returns and linefeeds delimiting lines, form feeds delimiting pages, and control-Z's delimiting the end of file), though all internal bit codes are transmitted. Receiving non-MS-DOS systems may well get confused when presented with nonstandard ASCII files. Files produced by EASYWRITER or Word Star, for example, may need preprocessing prior to transmission by commonly available "exporter" programs, to convert them to conventional ASCII formats. Spreadsheet data files, and dBASE II files need special formatting to be meaningful to non-MS-DOS recipients (though they can be transmitted between MS-DOSes with KERMIT-86's). Furthermore, those word processors storing formatting data at the end of the file, after the control-Z and before physical end (such as BLUE or Easy Writer), will need to be told to strip the formatting data, lest they confuse non-MS-DOS recipients.

10.2. Program Operation

KERMIT-86's prompt is "Kermit-86>". KERMIT-86 can run interactively to issue several commands, like this:

```
A>
A>kermit

MS DOS Kermit V2.25

Kermit-86>send foo.*
```

```
    informational messages about the files being sent

  Kermit-86>status

  various status informational data are displayed

  Kermit-86>receive

    informational messages about the files being recieved

  Kermit-86>exit
  A>
```

During interactive operation, you may use the help ("?") and recognition (ESC) features freely while typing commands. Command keywords may be abbreviated to their shortest prefix that sets them apart from any other keyword valid in that field.

10.3. MS DOS KERMIT Commands

MS DOS KERMIT implements a large subset of the local mode commands of "ideal" KERMIT. Not all of the following commands are available on all MS DOS systems, and some of the commands may work somewhat differently between DOS versions.

THE SEND COMMAND

Syntax: SEND filespec

The SEND command causes a file or file group to be sent from the MS-DOS to the other system. The filespec may contain a device designator, like A:, and the wildcard characters "*" and/or "=". The current release of Kermit-86 does not allow pathnames in this command.

If the filespec contains wildcard characters then all matching files will be sent, in directory search order (according to how your MS-DOS lists its directory contents). If a file can't be opened for read access, standard MS-DOS recovery procedures will be available (these may necessitate restarting Kermit).

SEND Command General Operation

Files will be sent with their MS-DOS filename and filetype (for instance FOO.TXT, no device or pathname). If you expect to be sending files whose names contain characters that would be illegal in filenames on the target system, and you know that Kermit on the target system does not have the ability to convert incoming filenames, you can copy and/or rename the file using MS-DOS commands prior to loading Kermit.

Each file will be sent as a sequence of eight bit bytes.

Once you give KERMIT-86 the SEND command, the name of each file will be displayed on your screen as the transfer begins; a packet count and retry summary will be displayed, and informational messages displayed as appropriate. If the file is successfully transferred, you will see "COMPLETED", otherwise there will be an error message. When the specified operation is done, the program will sound a beep.

If you notice a file being sent which you do not really want to send, you may cancel the operation immediately by typing either Control-X or Control-Z. If your are sending a file group, Control-X will cause the current file to be skipped, and KERMIT-86 will go on to the next file, whereas Control-Z will cancel sending the entire group and return you to KERMIT-86 command level. A Control-C cancels sending immediately and returns you to the Kermit-86 prompt.

THE RECEIVE COMMAND

Syntax: RECEIVE [filespec]

The RECEIVE command tells KERMIT-86 to receive a file or file group from the

other system. KERMIT simply waits for the file to arrive; this command is not to be used when talking to a KERMIT server (see GET).

If the optional filespec is provided, store the incoming file under that name. The filespec may include a device designator, or may consist of only a device designator. The incoming file is stored on the default or specified device (current directory in DOS 2.0). If no name was specified, the name from the incoming file header packet is used; if that name is not a legal MS-DOS file name, KERMIT-86 will delete illegal or excessive characters from the name.

If the optional filespec was provided, but more than one file arrives, the first file will be stored under the given filespec, and the remainder will be stored under their own names.

If the incoming file name already exists, and FILE-WARNING is set, KERMIT-86 will change the incoming name (and inform you how it renamed it) so as not to obliterate the pre-existing file.

If an incoming file does not arrive in its entirety, KERMIT-86 will normally discard it; 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 in your directory.

If a file begins to arrive that you don't really want, you can attempt to cancel it by typing Control-X; this sends a cancellation request to the remote Kermit. If the remote Kermit understands this request (this is an optional feature), it will comply; otherwise it will continue to send. If a file group is being sent, you can request the entire group be cancelled by typing Control-Z. If you type Control-C, you will be returned immediately to the Kermit-86> command level.

THE GET COMMAND

Syntax: GET remote-filespec

The GET remote-filespec command requests a remote KERMIT server to send the file or file group specified by remote-filespec. This command can be used only when KERMIT-86 is local, with a KERMIT server on the other end. This means that you must 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-86.

The remote filespec is any string that can be a legal file specification for the remote system; it is not parsed or validated locally. (A remote PC server will accept device names, but not path names in the filespec.) As files arrive, their names will be displayed on your screen, along with packet traffic statistics and error messages. You may type ^X to request that the current incoming file be cancelled, ^Z to request that the entire incoming batch be cancelled, and ^C to return immediately to the Kermit-86> prompt.

If the remote KERMIT is not capable of server functions, then you will probably get an error message back from it like "Illegal packet type". In this case, you must connect to the other Kermit, give a SEND command, escape back, and give a RECEIVE command.

THE BYE COMMAND

When running a local Kermit which is talking to a remote KERMIT server over a communications line, use the BYE command to shut down the server and log out its job, and exit from Kermit-86 to DOS.

THE FINISH COMMAND

Like BYE, FINISH shuts down the remote server. However, FINISH does not log out the server's job. You are left at Kermit-86 prompt level so that you can connect back to the job on the remote system.

THE LOGOUT COMMAND

The LOGOUT command is identical to the BYE command, except you will remain at Kermit-86 prompt level, rather than exit to DOS, so that you can establish another connection.

THE CONNECT COMMAND

Syntax: CONNECT

Establish an interactive terminal connection to the system connected to the currently selected communications port (COM1 or COM2) using full duplex echoing and no parity unless otherwise specified in previous SET commands. Get back to KERMIT-86 by typing the escape character followed by the letter C. The escape character is Control-] by default. When you type the escape character, several single-character commands are possible:

?	Help - prints the commands allowed (as below).
C	Close the connection and return to KERMIT-86.
S	Status of the connection.
B	Break signal is sent to the port (on the PC/XT you may also type CTRL-BREAK to send a BREAK).
^]	(or whatever you have set the escape character to be) Typing the escape character twice sends one copy of it to the connected host.

You can use the SET ESCAPE command to define a different escape character, and on some systems (including the PC and XT) you can SET BAUD to change the baud rate, and SET PORT to switch between COM1 and COM2

In the connect mode, you can communicate with your autodialer, control the communications line, hang it up, and the like. (E.g., typing +++ to a Hayes-like modem will allow you to follow that by dialing or hang-up commands, when in the connection state).

THE REMOTE COMMAND

The REMOTE keyword is a prefix for a number of commands. It indicates that the command is to be performed by the remote Kermit, which must be running as a server. Note that not all Kermit servers are capable of executing all these commands. In case you send a command the server cannot execute, it will send back a message to the effect that the command is unknown to it. If the remote can execute the command, it will send the results to your screen. Here are the REMOTE commands which KERMIT-86 may issue:

CWD [directory]	Change Working Directory on the remote host. Change the default source and destination area for file transfer.
DELETE filespec	Delete the specified file or files on the remote host. In response, the remote host should display a list of the files that were or were not successfully deleted.
DIRECTORY [filespec]	The remote host will provide a directory listing of the specified files. If no files are specified, then all files in the default area will be listed.
DISK [directory]	Provide a brief summary of disk usage in the specified area on the remote host. If none specified, the default or current area will be summarized.
HELP	The remote host tells what server functions it is capable of.
HOST [command]	Send the command to the remote host's command processor for execution.
TYPE filespec	Display the contents of the specified remote file or files on the screen.

THE SET COMMAND

Syntax: SET parameter [value]

Establish or modify various parameters for file transfer or terminal connection. You can examine their values with the STATUS command. The following parameters may be SET:

BACKARROW	Backarrow (backspace) key sends BACKSPACE or DELETE.
BAUD	Communications port line speed
BELL	The bell (beep) is normally sounded at the end of a transaction. SET BELL OFF may be used to silence the bell.
DEBUG	Mode
END-OF-LINE	Character to replace CR at end of packets
ESCAPE	Character for Kermit-86 attention during terminal connection
FILE-WARNING	Warn if an incoming filename would conflict with an existing file name, and attempt to construct a new unique name for it.
HEATH-19	Interpret Heath/Zenith-19 screen control codes.
IBM	Set up for communication with IBM mainframes: local echo (half duplex) during terminal emulation, line turnaround handshake during file transfer, and appropriate parity at all times.
INCOMPLETE	What to do with an incomplete file, KEEP or DISCARD.
LOCAL-ECHO	For terminal connection, OFF (remote echo, or full duplex) or ON (local echo, or half duplex)
PARITY	Character parity to use, NONE (the default), ODD, EVEN, MARK, or SPACE
PORT	RS232 port to use for terminal connection or file transfer, COM1 (the default) or COM2

SET BACKARROW

Syntax: SET BACKARROW state

The IBM PC keyboard does not have a key marked DELETE (RUBOUT) or BACKSPACE. DELETE and BACKSPACE are two different ASCII characters (ASCII 127 and ASCI 8 respectively), and one or the other of these characters is normally used by host systems for deleting the characters just typed. Some systems use BACKSPACE, some use DELETE. This command allows you to specify which character the backarrow key should transmit during terminal connection.

BACKSPACE	Backarrow (backspace) key transmits the backspace (BS) character, Control-H. CTRL-Backarrow sends DELETE.
DELETE	Backarrow (backspace) key transmits the delete (DEL, RUBOUT) character. CTRL-Backarrow sends BACKSPACE.

In all cases, CTRL-H sends BACKSPACE.

SET BAUD

Syntax: SET BAUD rate

Set terminal communications port speed to 300, 1200, 1800, 2400, 4800, 9600 or other common baud rates. The site default baud rate can be determined by the STATUS command immediately upon loading Kermit-86, and is displayed upon issuing of the CONNECT command.

SET BELL

Syntax: SET BELL state

ON Bell (beeper) sounds, at completion of transmissions and other times.

OFF Bell (beeper) remains silent.

SET DEBUG

Syntax: SET DEBUG state

ON Record the packet traffic on your terminal.

OFF Don't display debugging information (this is the default). If debugging was in effect, turn it off.

SET END-OF-LINE

Syntax: SET END-OF-LINE decimal number between 0 and 31

Change the character used at the end of outgoing packets to the character whose decimal ASCII value is given. The default is 13 (carriage return).

SET ESCAPE

Syntax: SET ESCAPE character Specify the control character you want to use to "escape" from remote connections back to KERMIT-86. The default is Control-].

SET FILE-WARNING

Syntax: SET FILE-WARNING option

Specify what to do when an incoming file has the same name as an existing file in the default directory of the default device. 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) so as not to destroy (overwrite) the pre-existing one. If OFF, the incoming file replaces the pre-existing file.

SET HEATH-19

Syntax: SET HEATH-19 option

ON Specifies that, in the connect state, incoming characters are to be examined for Heath/Zenith-19 terminal screen control commands (escape sequences), and if encountered, the commands are to be emulated on the PC screen. The Heath-19 codes are a superset of the popular DEC VT52 codes, so if your system does not support the Heath-19, you may tell your terminal type is VT52 (or one of the many VT52 compatibles). Heath-19 emulation is available on the IBM PC and XT.

OFF All incoming characters will be sent to the screen "bare", 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. Most non-IBM systems have their own screen control code interpreter built into DOS or firmware.

On the IBM systems, function keys and numeric keypad cursor control keys do not send characters when in the Heath-19 mode, unless the user has used a key redefinition package like ProKey.

SET IBM

Syntax: SET IBM option

Specify setup for communication with an IBM mainframe. ON sets appropriate parity (per options used to assemble Kermit at your site, MARK as distributed), local echo for CONNECT, and half-duplex line handshaking (XON line turnaround). OFF reestablishes full duplex, nonparity operation.

SET LOCAL-ECHO

Syntax: SET LOCAL-ECHO option

Specify mode for character echoing when in the CONNECT state. ON specifies that characters are to be echoed within Kermit (because neither the remote computer, nor the communications circuitry has been requested to echo). Generally IBM mainframes accessed directly (not via Telenet) will need this option (or the IBM option, q.v.) ON; generally most DEC sites and inter-PC communications will need it OFF. It is OFF by default, i.e. communication is assumed to be full duplex (remote echo).

SET PARITY

Syntax: SET PARITY keyword

The choices for SET PARITY are NONE (the default), ODD, EVEN, MARK, and SPACE. NONE means no parity processing is done, and the 8th bit of each character can be used for data when transmitting binary files.

You will need to SET PARITY to ODD, EVEN, MARK, or possibly SPACE when communicating with a system, or over a network, that requires or imposes character parity on the communication line. For instance, GTE TELENET requires MARK parity. If you neglect to SET PARITY when the communications equipment requires it, the symptom may be that terminal emulation works partially, and file transfer does not work at all.

If you have set parity to ODD, EVEN, MARK, or SPACE, then KERMIT-86 will request that binary files will be transferred using 8th-bit-prefixing. If the other side knows how to do 8th-bit-prefixing (this is an optional feature of the KERMIT protocol, and not all implementations of KERMIT have it), then binary files can be transmitted successfully. If NONE is specified, 8th-bit-prefixing will not be requested.

SET PORT

Syntax: SET PORT number

Specify the port number to use for file transfer or CONNECT, COM1 or COM2. This command lets you use a different asynchronous adapter, or to switch between two simultaneous remote sessions.

THE STATUS COMMAND

Report the status of parameters which can be modified by the SET commands.

10.4. Installation

Kermit-86 is written in 8086 Macro Assembler (ASM86), and assembled locally on the micro. Versions for the IBM PC (PC DOS) and the Heath/Zenith Z100 (MS DOS) are prepared from common source using conditional assembly switches similar to those in KERMIT-80. The IBM flag has site-dependent meaning. As shipped from Columbia, it means local echo during CONNECT, mark parity, and half duplex line handshaking using CTRL-Q as the turnaround character. If you need to install Kermit on your PC, and you do not have a Kermit floppy but you do have access to a mainframe computer with a copy of the IBM PC Kermit distribution, you should read this section.

Since the PC assembler is not provided with the minimum system, IBM PC users cannot be expected to have it. Assembler source plus the runnable version (.EXE) of Kermit are distributed [16], along with some special "bootstrap" files, described below.

The KERMIT.EXE file is converted by an assembler program on the PC, KFIX, which makes all bytes in the file printable by breaking each one up into two 4-bit "nibbles" and adding a constant. The result is a printable file called KERMIT.FIX. It is assumed that a copy of KERMIT.FIX is available to you on a mainframe computer. To download the file to the PC, two cooperating programs are run: a Fortran program, KSEND, on the mainframe and a Basic program, KGET, on the PC. These programs are very short; they are shown in their entirety below. KSEND reads a line at a time from KERMIT.FIX, types the line, and waits for a signal from KGET that it can send more data. KGET reads each line and converts the text back to the format of an executable (.EXE) file. Here's the procedure:

1. You should have a version of KGET on the PC and KSEND on the mainframe; if you don't have them, copy them (i.e. type them in, using an editor [17]) from the listings below.

2. Log in on the mainframe. This could be tricky if you have no terminal emulation facility on the PC. If you have the IBM asynchronous communication package, you can do this at low speeds (baud rates). If your PC has no terminal emulation facility, you'll have to use a real terminal to log in, and then switch the cable to the PC.

3. Compile KSEND.FOR on your mainframe, if it needs compiling. Define logical unit numbers 5 and 6 to be the controlling terminal, and logical unit number 7 to be KERMIT.FIX. On the DEC-20, for example:

```
@define 5: tty:
@define 6: tty:
@define 7: kermit.fix
```

   On a DECsystem-10, do something like:

```
.assign tty: 5:
.assign tty: 6:
.assign dsk: 7:
.rename for007.dat=kermit.fix
```

   On an IBM system under VM/CMS,

```
.filedef 5 term ( lrecl 64 recfm f
.filedef 6 term ( lrecl 64 recfm f
.filedef 7 disk kermit fix ( lrecl 62 recfm f perm
```

   Start KSEND on the mainframe. It will print a message, and then sit and wait for the PC to send back an OK; don't change any connectors until you see the message.

4. Escape back to the PC, or connect the PC to the mainframe. The PC's communication port should be connected with a cable to the modem that's connected to the mainframe (dialup, dedicated, switched, whatever hookup you normally have available for logging in on the mainframe from a terminal). If you were using a different terminal to log in to the mainframe, make sure the PC's communication port is set at the same speed.

5. Enter BASIC and run KGET on the PC. If KGET prints messages about i/o errors, run it again. If it still gets errors, reboot the PC

---

[16] The PC assembler's object (.OBJ) files are not printable, like CP/M hex files, so the Kermit-80 bootstrapping technique would not work here.

[17]

and try again. Once KGET is running, the transmission will begin. KGET will print each 62-character line of nibbles as it arrives from the mainframe. Each line should be the same length -- if you see a ragged edge, you can assume there has been a transmission error, and you should start the process again.

You'll also have to compile and load the KSEND program on the mainframe.

6. When transmission is complete, you'll see the BASIC "Ready" prompt again. Leave BASIC by typing SYSTEM. You should now have KERMIT.EXE on your PC. Try to run it. If you see the "Kermit-86>" prompt, try to CONNECT to the host mainframe and transfer some files. If Kermit doesn't run correctly, there may have been transmission errors, in which case you should start the process again from step 2 above.

KSEND.FOR - Mainframe Side of Bootstrap

This is the mainframe side, KSEND, in transportable Fortran (it should run on both DEC and IBM mainframes):

```
C     This Fortran program should be run on the mainframe in conjunction
C     with a Basic program on the IBM PC to transfer Kermit.Fix to the PC

      INTEGER A(62)

      WRITE(6,50)
50    FORMAT(' Ready to transfer data.....')

C     Get terminal handshake
100   READ (5,10,END=35)X
10    FORMAT(A1)

C     Get line from file
35    READ (7,20,END=90)A
20    FORMAT(62A1)

C     Write to tty
      WRITE (6,25)A
25    FORMAT(' ',62A1,';')
      GOTO 100
90    CONTINUE

C     Get final handshake
      WRITE (6,30)
30    FORMAT(' ',63('@'))
      STOP
      END
```

The final @'s tell KGET that the transmission is done. This works because the technique for forming KERMIT.FIX ensures that the file will contain no @'s.

KGET.BAS -- PC Side of Bootstrap

This is the PC side, KGET, in PC Basic. Note that the communication port is opened at 4800 baud (you could substitute any other speed).

```
  5  'Run this program on the PC in conjunction with a Fortran program on t
  6  ' mainframe to get Kermit to the PC
  7  ' Daphne Tzoar , December 1983
  8  ' Columbia University Center for Computing Activities
  9  '
  10 OPEN "com1:4800,n,8,1" AS #1        ' Clear the port status.
  20 CLOSE #1
  30 OPEN "com1:4800,n,8,1,cs,ds,cd" AS #1
  40 OPEN "KERMIT.EXE" FOR OUTPUT AS #2
  50 OK$ = "ok"
  60 PRINT#1,OK$             ' Tell host we're ready for data
  70 X$=INPUT$(63,#1)        ' Data plus semi-colon
  80 VALUE$ = LEFT$(X$,1)      'First char of input
  90 VALUE = ASC(VALUE$)
  100 IF VALUE = 64 OR VALUE = 192 GOTO 430    ' @ means we're done
  110 IF VALUE >= 176 AND VALUE <= 191 THEN GOTO 140   ' Kill all illegal c
  120 IF VALUE >= 48 AND VALUE <= 63 THEN GOTO 140
  130 X$ = MID$(X$,2) : GOTO 80
  140 IF VALUE <> 174 GOTO 210     ' Not a dot (for read) - don't worry
  150 TWO$ = MID$(X$,2,1)          ' Look at char after the dot.
  160 TWO = ASC(TWO$)
  170 IF TWO >= 176 AND TWO <= 191 THEN GOTO 210    ' It's ok.
  180 IF TWO >= 48 AND TWO <= 63 THEN GOTO 210
  190 X$ = MID$(X$,3)         ' Kill the char
  200 GOTO 80
  210 SIZ = LEN(X$)           ' How much input was actual data
  220 READIN = 64 - SIZ
  225 IF READIN = 0 GOTO 260
  230 XTWO$=INPUT$(READIN,#1)     ' Get rest of data
  240 X$ = X$ + XTWO$ : X$ = LEFT$(X$,62)
  250 PRINT X$         ' Optional - use this line to follow the transmissio
  260 GOSUB 290
  270 PRINT#2,X$;      ' Put data to the file.
  280 GOTO 60
  290 ' GET TWO CHARS, SUBTRACT SPACE (20 HEX) FROM EACH, AND COMBINE
  300 ' TO ONE DIGIT.
  310 FOR A = 1 TO 31
  320 Y$ = MID$(X$,A,1)
  330 Z$ = MID$(X$,A+1,1)
  340 YNUM = ASC(Y$) : ZNUM = ASC(Z$)
  350 IF YNUM > 127 THEN YNUM = YNUM - 128    ' Turn off hi bit if on
  360 IF ZNUM > 127 THEN ZNUM = ZNUM - 128
  370 YNUM = YNUM -48 : ZNUM = ZNUM -48       ' Subtract the space
  380 XNUM = (16 * YNUM) +ZNUM
  390 NEWCHR$ = CHR$(XNUM)
  400 X$ = MID$(X$,1,A-1) + NEWCHR$ + MID$(X$,A+2)
  410 NEXT A
  420 RETURN
  430 PRINT  " [All done.]"
  440 CLOSE #1,#2                          ' Clean up.
  450 END
```

If you already have a working Kermit on your PC and you want to get a new one, you should use Kermit itself to transfer the KERMIT.FIX file. Once you have the new KERMIT.FIX on your PC disk:

1. Rename KERMIT.EXE to something else, so you'll still have it in case something goes wrong.

2. Get or copy the program KEXE from the mainframe. Alternatively, you may modify KGET as follows:
   a. Remove lines 10 and 20.
   b. Change line 30 to

```
30   OPEN "KERMIT.FIX" FOR INPUT AS #1
```

   c. Remove line 60, since we're not handshaking with a remote host

any more.
   d. In line 70, change "63" to "62".
   e. Remove line 250, since there's no need to monitor a transmission line.
   f. Change line 280 from "GOTO 60" to "GOTO 70".

   Save the modified KGET under a new name, say KEXE.BAS, and run it. It will end with some error like "Input past end in 70", which just means it came to the end of file (of course, you could avoid this error by trapping it, but no harm is done in any case).

3. You should now have a new, working version of KERMIT.EXE on your PC disk.

10.5. Adding Support for New Systems

MS DOS Kermit supports many different systems. Like CP/M-80 KERMIT, this support was added to the program piecemeal, at many sites, using conditional assembly. However, before allowing the program to grow into a complicated monolith like CP/M-80 KERMIT, we have broken the program up into separate modules, with system dependencies isolated into separate modules, consisting of compact collections of low-level primitives for console and port i/o.

The last monolithic (single source file) release of MS DOS Kermit was 1.20. To this and earlier versions was added support for systems like the Seequa Chameleon, the HP-150, the Victor 9000, the Heath/Zenith 100, and others. As time permits, support for these systems will be integrated with the new modular version. Meanwhile, implementations based on these old versions 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 REMOTE command.

- No 8th-bit prefixing.

To install support for a new system, you would copy the system-dependent modules for terminal emulation and port and console i/o, modify them to suit the requirements of your machine, and rebuild the program. In many cases, a "generic" MS DOS Kermit will run as-is on new systems. The generic version accomplishes all its port and console i/o through DOS calls, and does no terminal emulation -- many systems do not need terminal emulation because they have terminal firmware built in.

Details to be filled in...