VAX/VMS KERMIT-32 USER GUIDE

Version 3.3.126

Christine M. Gianone and Frank da Cruz
Columbia University Center for Computing Activities
New York, New York 10027

July, 1990

1. VAX/VMS KERMIT

Authors: Robert C. McQueen, Nick Bush, Stevens Institute of Technology;
Jonathan Welch, University of Massachusetts;
Burt Johnson, Diversified Computer Systems, Inc.

Language: Bliss-32

Documentation: C. Gianone, F. da Cruz, Columbia University
with thanks to the program's authors

Version: `3.3.126`

Date: July, 1990

VAX/VMS Kermit-32 Capabilities At a Glance:

Local operation:	Yes
Remote operation:	Yes
Transfers text files:	Yes
Transfers binary files:	Yes
Wildcard send:	Yes
Long packets:	Yes
Sliding windows:	No
Attribute packets:	No
File transfer interruption:	Yes
Filename collision avoidance:	Yes
Timeouts:	Yes
8th-bit prefixing:	Yes
Repeat character compression:	Yes
Alternate block check types:	Yes
Communication settings:	Yes
Transmit BREAK:	Yes
IBM mainframe communication:	Yes
Transaction logging:	Yes
Session logging (raw capture):	Yes
Debug logging:	Yes
Raw transmit:	Yes
Act as server:	Yes
Talk to server:	Yes
Advanced commands for servers:	Yes
Local file management:	Yes
Initialization file:	Yes (`VMSKERMIT.INI`)
Command Macros:	No
Script programming language:	No
International Character Sets:	No

Kermit-32 is a program that implements the Kermit file transfer protocol for the Digital Equipment Corporation VAX series computers under the VAX/VMS operating system. It is written in BLISS-32 and MACRO-32, with sources for all BLISS modules also available as MACRO-32 sources. Kermit-32 should run on any VAX/VMS system from version 4.0 on (Version `3.1` of Kermit-32 is the last version that runs under pre-4.0 releases of VMS).

The first section of this chapter will describe the things you need to know about the VAX/VMS file system and how Kermit-32 uses it. The second section describes the special features of Kermit-32. The final section contains information of interest to those who need to install Kermit-32 on a system.

1.1. The VAX/VMS File System

The two main items of interest of the VAX/VMS file system (for the Kermit user) are the format of file specifications and the types of files and file data.

VAX/VMS File Specifications

VAX/VMS file specifications are of the form

```
NODE::DEVICE:[DIRECTORY]NAME.TYPE;VERSION
```

Under version 4.0 and later of VMS, NAME, TYPE and each item in DIRECTORY may be up to 39 characters long, and may contain alphanumeric characters plus underscore. Under earlier versions, NAME and DIRECTORY could be at most 9 characters each and TYPE could be at most 3.

VERSION is a decimal number indicating the version of the file (generation). DEVICE may be either a physical or logical device name. If it is a logical name, it may be up to 63 characters long and may contain alphanumeric characters plus dollar signs and underscores. NODE may be either a logical name which translates to a DECnet node name or a physical DECnet node name. In either case, access information can be included (see the DECnet-VMS User's guide for more information). The node name is not normally present, since most files are accessed on the same node where the user's job is running. The version number is not normally given (in fact, should not normally be given). When device and/or directory are not supplied, they default to the user's current default device and directory. Therefore, `NAME.TYPE` is normally all that is needed to specify a file on the user's default device and directory. This is also all that Kermit-32 will normally send as the name of a file being transferred.

The node field specifies the name (and access information) for the DECnet node where the file is located. Kermit-32 does not transmit the node field to the target system, but will attempt to honor a node field in an incoming file name.

The device field specifies a physical or "logical" device upon which the file is resident. The directory field indicates the area on the device, for instance the area belonging to the owner of the file. Kermit-32 does not normally transmit the device or directory fields to the target system, but will attempt to honor device or directory fields that may appear in incoming file names. It will not create new directories, however, so any directory specified in an incoming filename must already exist.

The name 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.FOR` is the source of a Fortran program named FOO; `FOO.OBJ` might be the relocatable object module produced by compiling `FOO.FOR`; `FOO.EXE` could an executable program produced by LINKing `FOO.OBJ`, and so forth.

VAX/VMS 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, including no characters at all; a "`%`" matches any single character. Here are some examples:

- `*.FOR` All files of type `FOR` (all Fortran source files) in the default 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 at least one X.
- `%.*` All files whose names are exactly one character long.
- `*.%%*` All files whose types are at least two characters 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.

Text Files and Binary Files

The file system used by VAX/VMS provides for a large number of attributes to be associated with each file. These attributes provide some indication of whether the file is a text file, or is some other type of non-text data. The two major attributes that affect VMS Kermit are the record type and record attribute. The record type describes how logical records are stored in the file. Records may be of some fixed length (specified by another attribute), or variable length (specified within each record), or stream (implying that records -- if there are any -- are separated by control characters within the data). The record attributes describe how the breaks between records are to be treated. For example, a record attribute of implied carriage return means that any program reading the file with intentions of printing it should add a carriage return / linefeed sequence between each record. Other record attributes include FORTRAN carriage control and print file format.

The most common method of storing text in a file under VAX/VMS is to store one line of text per record (variable length records), with a carriage return / linefeed sequence implied by the end of the record (implied carriage return). This is the method Kermit-32 uses to store files it receives when using FILE TYPE ASCII (text). Other formats are also used to store text under VAX/VMS, but the one used by Kermit-32 is the only one that is handled correctly by all known utility programs under VAX/VMS. Also, most programs which work with text files (the editor EDT, for example) place some limit on the length of the lines which can be handled. Typically this is 255. Kermit-32 can write text files with up to 4095 characters on a line, which means a text file from another system may be transferred and stored correctly by Kermit-32, but may still be unusable by certain VAX/VMS programs.

Certain PC applications may create text files with lines even longer than Kermit-32's maximum. Typical examples are the ASCII export procedures of database, spreadsheet, and CAD packages. If you try to send such a file to Kermit-32, the transfer will fail with a message:

```
%KERMIT32-E-REC_TOO_BIG, Record too big for KERMIT's internal buffer
```

If this happens, you can SET FILE TYPE BINARY on the VAX before transferring the file to it. You should still be able to use the file as a text file, with the above proviso about record length.

There is no standard format for storing binary files. In general, any record format with no record attributes can be used for binary files. Since programs which work with binary files under VAX/VMS expect to see some particular format, more information is needed for transfer of binary files than for text files. The current version of Kermit-32 is not capable of transferring all types of binary files which were created on a VAX/VMS system to another system and retrieving them intact, nor is it capable of transferring all of types binary files created on a VAX/VMS system to another VAX/VMS, P/OS, or RSX-11M/M+ system intact. However, certain formats of binary files can be transferred, and binary files from some other systems may be transferred to a VAX and recovered intact. See the section on the SET FILE command for details.

Using two programs supplied with Kermit-32, it is possible to transfer almost any type of file between VAXes, or between a VAX and a P/OS or RSX-11M/M+ system. The VMSHEX program converts a binary file to text (using a variation on Intel hex format). The resulting file can be transferred as an ordinary text file, and finally "dehexified" on another VMS system using the VMSDEH program to reproduce the original file with all the attributes intact. Since these text files are about twice the size of the original binary files, the transfers take longer. On the plus side, the text versions of the files can be transferred to any system with a Kermit and still retrieved intact. They can also be transferred over 7-bit data paths without any problems. The Kermit-32 installation procedure (described later) makes use of hexified versions of the Kermit-32 binary executable file and VMSDEH to restore it a binary `.EXE` file, or task image.

Using the VAX to Archive Microcomputer Files

You can use Kermit to send textual files from a microcomputer or any 8-bit computer system to VAX/VMS with no special provisions, since Kermit-32 stores incoming files as text files (variable length records with implied carriage returns) unless it is explicitly told otherwise. But Kermit-32 has no automatic way of distinguishing an incoming binary file from an incoming text file. You must inform Kermit-32 that a file it is about to receive is to be stored as a binary file. This is done using the SET FILE TYPE BINARY command. This instructs Kermit-32 to store the data it receives in the file without checking for carriage return, line feed sequences. The file it creates will be variable record length, with no record attributes. Each record will contain 510 bytes of data, except the last, which will contain whatever amount is left before the end of the file. This allows Kermit-32 to correctly return exactly the data it was sent when the file is returned to the original system.

Note that because of the need to use a different file type for binary files, it is not possible to use a "wildcard send" command to send a mixture of text and binary files to a VAX system unless the text files are not intended for use on the VAX; rather, you must send all text files with Kermit-32's file type set to ASCII, and all binary files with the file type set to binary.

Once you get the foreign file into the VAX system, stored with the correct file type, you need take no special measures to send it back to its system of origin. This is because Kermit-32 honors the record type and attributes of the file as it is stored on the VAX. In fact, SET FILE TYPE BINARY, ASCII, or FIXED only affects how Kermit-32 receives files -- it does not affect how Kermit-32 transmits files.

Files Kermit-32 Cannot Handle

The Kermit protocol can only accommodate transfer of *sequential* files, files which are a linear sequence of bytes (or words).

Some files on a VAX/VMS system are not sequential, and cannot be successfully sent or received by Kermit-32. These are mainly indexed data files, but can also include other files which require more than just the data in the file to be completely reconstructed. External control information and file attributes are not transmitted. However, *any* VMS file can be transferred with Kermit if it has been "hexified" with VMSHEX.

1.2. Program Operation

If a system-wide foreign command "kermit" is defined for Kermit-32 (see section on installation), then you can run the program by just typing its name:

```
$ kermit
```

If you get a message like:

```
%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
```

then Kermit has not been installed properly on your VMS system. If you know where the `KERMIT.EXE` file is stored (for example, in the `SYS$SYSTEM:` area) then you can define a "kermit" command for yourself by including a line like this in your `LOGIN.COM` file:

```
kermit :== $sys$system:kermit.exe
```

When you invoke Kermit by name only, it will enter interactive prompting mode, and allow you to type repeated commands until you exit with the EXIT command. Kermit-32's normal prompt is "`Kermit-32>`".

Kermit-32 will also accept a single command on the command line, like this:

```
$ Kermit send foo.bar
```

In this case, the program will exit immediately after executing the single command.

In either case, Kermit reads and executes commands from its "initialization file", `VMSKERMIT.INI`, if any, in your login directory before it executes any other commands.

Kermit-32 will try to open the file `VMSKERMIT` with a default filetype of `.INI`. If the logical name VMSKERMIT exists then an attepmt to open the file pointed to by the value of that logical name will be made instead. (For example, some sites might wish to set site-wide default parameters in a system-wide Kermit-32 initialization file, and so they might define the system-wide logical name VMSKERMIT to point at such a file. The last command in this file could be `TAKE SYS$LOGIN:VMSKERMIT.INI` to "chain" to the user's initialization file.

Command keywords may be abbreviated to their shortest prefix that sets them apart from any other keyword valid in that field.

1.3. Conditioning Your Job for Kermit

Kermit-32 does as much as it can to condition your line for file transfer. It saves all your terminal settings, and restores them after use. However, there are some sources of interference over which Kermit-32 has no control. In particular, messages issued by other processes in your job could become mingled with Kermit packets and slow things down or stop them entirely. This is a fairly rare occurence and can be easily avoided by not running any other process which wishes to perform I/O to your terminal while you are running Kermit-32.

Normally, when Kermit-32 is run, it assumes you wish to use it in remote mode and perform file transfers over the terminal line which controls your job. This can be overridden, however, by defining a logical name which equates to some other terminal line in the system. The default terminal line to be used for file transfers is determined by the first of the following logical names which translates to a terminal line which is available for use by your process: KER$COMM, SYS$INPUT, SYS$OUTPUT, and SYS$COMMAND. If none of these logical names translate to an available terminal line, there is no default terminal line and a SET LINE command must be used before any transfer command is performed. Note that this is the typical case in a batch job.

Kermit-32 will also default the type of parity to be used on the communication line to that which is set on its default terminal line when it is started. This means that if all communication at a site is normally done using even parity (for example), Kermit-32 will also use even parity. If you need to use another kind of parity, use the SET PARITY command to change it.

There are two things to keep in mind when using Kermit-32 in local mode (where the file transfers are done over a different terminal line from where commands are typed):

- Under VAX/VMS, every terminal line has an owner UIC and protection code associated with it. This UIC and protection is used to determine who can allocate (and therefore use) the terminal line when they are not logged in on that line. Therefore, in order for Kermit-32 to be able to perform file transfers over a terminal line other than the one on which you are logged in, the field of the protection code for the terminal which applies to your job (based on your UIC and the owner UIC of the terminal) must allow your job access to the terminal. You may need to request your system manager to change the protection for a terminal line to allow you to use it with Kermit-32 in local mode. See the section on Installation for details.
- Terminal lines which have been declared as modem control lines will have the phone "hung up" whenever the terminal line becomes free (deallocated). This means that if you do not use the DCL ALLOCATE command to allocate the terminal line to your job before entering Kermit-32, exiting Kermit-32 will cause the terminal line to "hang up" the modem. If you do wish to get to DCL after having used Kermit-32 to connect a modem control line which you do not have allocated, you can use the PUSH command to spawn a subprocess running DCL, so that Kermit will keep the connection active.

1.4. Kermit-32 Commands

Kermit-32 has the following commands:

@	synonym for "take".
BYE	to remote server.
CONNECT	as terminal to remote system.
EXIT	from Kermit-32.
FINISH	Shut down remote server.
GET	remote files from server.
HELP	with Kermit-32.
LOCAL	prefix for local file management commands.
LOG	remote terminal session.
LOGOUT	remote server.
PUSH	to DCL command level.
QUIT	from Kermit-32.
RECEIVE	files from remote Kermit.
REMOTE	prefix for remote file management commands.
SEND	files to remote Kermit.
SERVER	mode of remote operation.
SET	various parameters.
SHOW	various parameters.
STATUS	about most recent file transfer.
TRANSMIT	Transmit (upload) a file with no error checking.
TAKE	Kermit-32 commands from a file.

1.4.1. Commands for File Transfer

Kermit-32 provides the standard SEND, RECEIVE, and GET commands for transferring files using the Kermit protocol.

The SEND Command

Syntax: `SEND` *filespec*

The SEND command causes a file or file group to be sent from the VAX to the other system. If *filespec* contains wildcard characters then all matching files will be sent, in alphabetical order (according to the ASCII collating sequence) by name. If *filespec* does not contain any wildcard characters, then the single file specified by *filespec* will be sent.

Only the most recent generation of a file is sent unless the file specification includes specific or wild generation numbers.

Files will be sent with at least their VAX/VMS file name and type (for instance `FOO.BAR`). If a SET FILE NAMING FULL command has been given, Kermit-32 will also send the device name, directory name and version number (for instance `USER$DISK:[JOE]FOO.BAR;25`). If a SET FILE NAMING UNTRANSLATED command has been given, Kermit-32 will send the file name, type and version number (for instance `FOO.BAR;25`). If a SET FILE NAMING NORMAL_FORM command has been given (this is the initial default), Kermit-32 will only send the file name and type.

Each file will be sent according to the record type and attributes recorded in its file descriptor. Kermit-32 attempts to translate all text file record formats (including those with FORTRAN or print carriage control) to a format usable on any system. Note that there is no need to set the FILE TYPE parameter for sending files, since Kermit-32 always uses the information from the file descriptor to determine how to send the file.

If communication line parity is being used (see SET PARITY), Kermit-32 will request that the other Kermit accept a special kind of prefix notation for files that contain 8-bit data. This is an optional Kermit protocol feature, supported by most modern Kermit programs. If the other Kermit does not agree to use this feature, binary files cannot be sent correctly. This includes executable programs (like `.EXE` files, CP/M `.COM` files), relocatable object modules (`.OBJ` files), as well as any text file containing characters with the eighth bit on.

Kermit-32 will also ask the other Kermit whether it can handle a special prefix encoding for repeated characters. If it can, then files with long strings of repeated characters will be transmitted very efficiently. Columnar data, highly indented text, and binary files are the major beneficiaries of this technique.

If you're running Kermit-32 in local mode, for instance dialing out from a VAX to another system using an autodialer, you should have already run Kermit on the remote system and issued either a RECEIVE or a SERVER command. Once you give Kermit-32 the SEND command, the name of each file will be displayed on your screen as the transfer begins. If the file is successfully transferred, you will see "`[OK]`", otherwise there will be an error message.

During local operation, you can type Control-A at any point during the transfer to get a brief status report. You may also type Control-X or Control-Z to interrupt the current file or file group.

The RECEIVE Command

Syntax: `RECEIVE [`*filespec*`]`

The RECEIVE command tells Kermit-32 to receive a file or file group from the other system. If only one file is being received, you may include the optional *filespec* as the name to store the incoming file under; otherwise, the name is taken from the incoming file header. If the name in the header is not a legal VAX/VMS file name, Kermit-32 will normally replace the illegal characters with "X" (see SET FILE NAMING NORMAL_FORM).

If an incoming file has the same name as an existing file, Kermit-32 just creates a new version of the same name and type, for instance `FOO.BAR;3, FOO.BAR;4`.

Incoming files are stored with the prevailing file type, ASCII by default, which is appropriate for text files. If you are asking Kermit-32 to receive binary files from a microcomputer or other 8-bit system, you must first type SET FILE TYPE BINARY. Otherwise, an error may occur when receiving the file, or carriage return / linefeed sequences will be added to the file, making it useless when sent back to the system of origin.

If parity is being used on the communications line, then 8th-bit prefixing will be requested. If the other side cannot do this, binary files cannot be transferred correctly. If parity is being added externally to Kermit and VMS (for example by some kind of communication device, or a public data network like Telenet) then you must inform Kermit-32 about it using the SET PARITY command, or else Kermit-32 will not know that it has to do 8th-bit prefixing, and the file transfer will fail.

If an incoming file does not arrive in its entirety, Kermit-32 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 you are running Kermit-32 in local mode, you should already have issued a SEND command to the remote Kermit, and then escaped back to Kermit-32. As files arrive, their names will be displayed on your screen. You can type Control-A during the transfer for a brief status report.

If a file arrives 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 (not all implementations of Kermit support this 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.

The GET Command

Syntax: `GET [`*remote-filespec*`]`

The GET 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-32 is in local mode, with a Kermit server on the other end of the line specified by SET LINE. This means that you must have CONNECTed to the other system, logged in, run Kermit there, issued the SERVER command, and escaped back to the VAX.

The remote filespec is any string that can be a legal file specification for the remote system; it is not parsed or validated locally. Any leading spaces before the remote filespec are stripped, and lower case characters are raised to upper case.

As files arrive, their names will be displayed on your screen. As in the RECEIVE command, you may type Control-A to get a brief status report, Ctrl-X to request that the current incoming file be cancelled, Ctrl-Z to request that the entire incoming batch be cancelled.

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 STATUS Command

Give statistics about the most recent file transfer.

The PUSH Command

Syntax: `PUSH`

Spawn a DCL subprocess, to which you may issue any DCL commands. Type LOGOUT to return to Kermit-32.

The TAKE Command

Syntax: `TAKE` *file-spec* `[ /DISPLAY ]`

Where 'file-spec' is any normal VAX/VMS file specification. If file-spec does not specify a file-type Kermit-32 will supply a default of `.COM`. The /DISPLAY option causes the commands read from the file to be displayed on the user's terminal.

The TAKE command tells Kermit-32 to execute commands from the specified file. You may also use the VMS notation "`@`" instead of TAKE to specify a command file.

If it exists, the file `VMSKERMIT.INI` (or, if the logical name VMSKERMIT is defined, whatever file it points to) is automatically taken upon program startup.

1.4.2. Server Operation

The SERVER Command

The SERVER command puts a remote Kermit-32 in "server mode", so that it receives all further commands in packets from the local Kermit. The Kermit-32 server is capable (as of this writing) of executing the following remote server commands: SEND, GET, FINISH, BYE, REMOTE DIRECTORY, REMOTE CWD, REMOTE SPACE, REMOTE DELETE, REMOTE TYPE, REMOTE HELP, REMOTE COPY, REMOTE RENAME, REMOTE SEND_MESSAGE, REMOTE WHO, and REMOTE HOST.

Any nonstandard parameters should be selected with SET commands before putting Kermit-32 into server mode, in particular the file type. The Kermit-32 server can send all files in the correct manner automatically. However, if you need to ask Kermit-32 to receive binary files you must issue the SET FILE TYPE BINARY command before putting it into server mode, and then you must only send binary files. You cannot send a mixture of text files and 8-bit binary files to a Kermit-32 server unless the files are not for use on the VAX.

Commands for Servers

When running in local mode, Kermit-32 allows you to give a wide range of commands to a remote Kermit server, with no guarantee the that the remote server can process them, since they are all optional features of the protocol. Commands for servers include the standard `SEND`, `GET`, `BYE`, `LOGOUT` and `FINISH` commands, as well as the `REMOTE` command.

Syntax: `REMOTE` *command*

Send the specified command to the remote server. If the server does not understand the command (all of these commands are optional features of the Kermit protocol), it will reply with a message like "Unknown Kermit server command". If does understand, it will send the results back, and they will be displayed on the screen. The REMOTE commands are:

COPY *filespec*
: Copy file. The server is asked to make a copy of the specified file. Kermit-32 will prompt for the new file name on a separate line. Both filespecs must be in the correct format for the remote system. Kermit-32 does not parse or validate the file specifications. Any leading spaces will be stripped and lower case characters converted to upper case. Note that this command simply provides for copying a file within the server's system - it does not cause a file to be transferred.

CWD [*directory*]
: Change Working Directory. If no directory name is provided, the server will change to the default or home directory. Otherwise, you will be prompted for a password, and the server will attempt to change to the specified directory. The password is entered on a separate line, and does not echo as you type it. If access is not granted, the server will provide a message to that effect. Note that while not all server Kermits require (or accept) a password to change the working directory, Kermit-32 will always ask for one when a directory name is provided.

DELETE *filespec*
: Delete the specified file or files. The names of the files that are deleted will appear on your screen.

DIRECTORY [*filespec*]
: The names of the files that match the given file specification will be displayed on your screen, perhaps along with size and date information for each file. If no file specification is given, all files from the current directory will be listed.

DISK_USAGE [*directory*]
: Display information about disk usage in the given directory (or by the given user). If no directory is provided, disk usage information is provided for the current working directory (or user). This is the same as the REMOTE SPACE command.

EXIT
: Requests the server to leave Kermit, allowing the terminal to be used for normal commands.

FINISH Requests the server to return to the Kermit prompt, allowing statistics to be obtained about the transfers.

HELP [*topic*] Provide information about the given topic. If no topic is given, provide a list of the functions that are available from the server. Some servers may ignore the topic and always display the same information.

HOST [*command*] Pass the given command to the server's host command processor, and display the resulting output on your screen.

LOGIN *user-id* Supply information to the server Kermit to indicate what user-id, account and password are to be used. The server Kermit may use this to validate the user's access to the system as well as for billing purposes. It may also use this information to provide the user with access to files on its system.

LOGOUT Request the server to exit Kermit and logout its job (or process). This command is identical to the LOGOUT command.

RENAME *filespec* Change the name on the specified file (or files). Kermit-32 will prompt for the new file specification on the next line. Both file specifications must be valid for the server's system.

SEND_MESSAGE *destination-address*
Request the server to send a single line message to the specified destination address (which might be a user-id, terminal designator, or some other item, depending upon the server Kermit). Kermit-32 will prompt for the single line message on the next line.

SPACE [*directory*] Display information about disk usage in the given directory (or by the given user). If no directory is provided, disk usage information is provided for the current working directory (or user). This is the same as the REMOTE DISK_USAGE command.

STATUS Display information about the status of the server.

TYPE *filespec* Display the contents of the specified file on your screen.

WHO [*user-id*] Display information about the given user. If no user-id is given, display information about the currently active users. Kermit-32 will prompt for options for selecting what information to display and/or formatting parameters. The format of both the user-id and the options are dependent upon the server Kermit.

1.4.3. Commands for Local File Management

Syntax: `LOCAL [`*command*`]`

Execute the specified command on the local system -- on the VAX/VMS system where Kermit-32 is running. These commands provide some local file management capability without having to leave the Kermit-32 program. These commands are very similar to the REMOTE commands in function and syntax. They are all executed locally, and are available when Kermit-32 is either local or remote. The arguments to these commands are the same as the arguments expected from the user Kermit when Kermit-32 is processing a command in server mode.

COPY *filespec* Make a copy of the given file (or files). Kermit-32 will prompt for the new file specification. The command is actually performed by using the DCL COPY command (COPY/LOG *old-file new-file*), and any options which are valid on the DCL COPY command may be included.

CWD [*directory*] Change working directory, or, in VAX/VMS terminology, change the default device/directory. This command takes the same arguments as the DCL SET DEFAULT command (i.e., a device and directory, only a directory, or only a device). If no argument is given, the default device and directory are reset to that in effect when Kermit-32 was run. The new default device and directory will be typed out.

DELETE *filespec* Delete the specified file or files. This command is performed by using the DCL DELETE command (DELETE/LOG *filespec*). Therefore, any options which are valid on the DCL DELETE command may be included.

DIRECTORY [*filespec*]
Provide a directory listing of the specified files. This command is performed by using the DCL

DIRECTORY command (DIRECTORY *filespec*), so any options valid for the DCL DIRECTORY command may be included.

DISK_USAGE [*uic*]
: Display disk usage information for the given UIC. If no UIC is given, display disk usage information for the process UIC. This command is performed by using the DCL SHOW QUOTA command (SHOW QUOTA or SHOW QUOTA/USER=*uic*).

HELP
: Display the help message describing the server commands which are available.

HOST *DCL-command*
: Perform the given DCL command. The command should not perform any action which will require more input. Any output resulting from the command will be typed on the terminal.

RENAME *filespec*
: Change the name of the specified file. Kermit-32 will prompt for the new name on the next line. This command is performed by using the DCL RENAME command (RENAME/LOG *old-file new-file*), so any options which are valid on the DCL RENAME command may be included.

SEND_MESSAGE *terminal-name*
: Send a single line message to the given terminal. Kermit-32 will prompt for the message on the next line. Since this command is performed using the DCL REPLY command

```
REPLY/TERMINAL=terminal-name "message"
```

OPER privileges are needed to perform it.

TYPE *filespec*
: Display the contents of the specified file or files at your terminal. Each file will be preceded by its name in angle brackets.

1.4.4. The CONNECT Command

Syntax: `CONNECT [`*terminal-name*`]`

Establish a terminal connection to the system connected to the terminal line specified here or in the most recent SET LINE command, using the currently set communication parameters (local-echo, parity, etc). Get back to Kermit-32 by typing the escape character followed by the letter C. The escape character is Control-Rightbracket (`^]`) by default. When you type the escape character, several single-character commands are possible:

- `B` Send a BREAK signal.
- `C` Close (but do not hang up) the connection and return to Kermit-32.
- `Q` If a session log is active, temporarily Quit logging.
- `R` Resume logging to the session log.
- `S` Show status of the connection.
- `0` Send a null character.
- `?` List all the possible single-character arguments.
- `^]` (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 SET PARITY, and SET LOCAL_ECHO to change those communication-line-oriented parameters. Type the SHOW LINE command for information about your current communication settings.

Kermit-32 does not have any special autodialer interface. It assumes that the connection has already been made and the line assigned. If the line has an autodialer attached to it, then you can type commands to the autodialer after you CONNECT.

1.4.5. The SET and SHOW Commands

The SET Command

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

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

BLOCK_CHECK	Packet transmission error detection method
DEBUGGING	Record or display state transitions or packets
DELAY	How long to wait before starting to send
ESCAPE	Character for terminal connection
FILE	For setting file parameters like file type
HANDSHAKE	For establishing half duplex line turnaround handshake
IBM_MODE	For communicating with an IBM mainframe
INCOMPLETE_FILE	What to do with an incomplete file
LINE	Terminal line to use for file transfer or CONNECT
LOCAL_ECHO	For terminal connection, ON or OFF
MESSAGE	The type of typeout to be done during transfers
PARITY	Character parity to use
PROMPT	Change the program's command prompt
RECEIVE	Various parameters for receiving files
REPEAT_QUOTE	Character to use for repeat compression
RETRY	How many times to retry a packet before giving up
SEND	Various parameters for sending files
TRANSMIT	Control TRANSMIT command echo and delay

SET DEBUGGING

Syntax: `SET DEBUGGING` *options*

Record the packet traffic, either on your terminal or in a file. Some reasons for doing this would be to debug a version of Kermit that you are working on, to record a transaction in which an error occurred for evidence when reporting bugs, or simply to vary the display you get when running Kermit-32 in local mode. Options are:

ON Display each incoming and outgoing packet (lengthy).

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

The debugging information is recorded in the file specified by the most recent LOG DEBUGGING command.

SET ESCAPE

`SET ESCAPE` *octal-number*

Specify the control character you want to use to "escape" from remote connections back to Kermit-32. The default is 35 (Control-]). The number is the octal value, 1 to 37 (or 177), of the ASCII control character you want to use, for instance 2 is Control-B.

SET FILE

Syntax: `SET FILE` *parameter keyword*

Establish file-related parameters:

BLOCKSIZE *number*

Specify the *record size* for incoming files when the file type is set to BINARY, FIXED, or BLOCK. Note that "blocksize" is a misnomer, but one which is commonly used in VMS. All VMS disk files have a true blocksize of 512 bytes. The Kermit "blocksize" (as well as the blocksize referred to in many VMS commands, like BACKUP, and help files) is really the record size.

TYPE *keyword*

How Kermit-32 should treat and store the file that is being sent to it, i.e. that Kermit-32 is *receiving*, and (in the case of FILE TYPE BLOCK only) how it is to read a file it is *sending* from disk. The choices are ASCII, BINARY, BLOCK, and FIXED. The BINARY, BLOCK, and FIXED types use a default record size (described below) which may be overriden, for received files only, with the SET FILE BLOCKSIZE command. Because the VMS file system is so complex, and because files created by different applications can have different characteristics, you might have to experiment with different values for the SET FILE TYPE and SET FILE BLOCKSIZE commands before you find the one that works right for you.

ASCII This is the default file type. Incoming files are stored as standard VAX/VMS text files with variable length records and carriage return / line feed sequences implied between records (that is, with the CR carriage control record attribute). This is the format preferred by most utility programs under VAX/VMS. A fatal error occurs if any line is more than 4096 characters long. Note that incoming lines are only terminated by carriage return, line feed sequences. A carriage return that is not followed by a line feed or a line feed that is not preceded by a carriage return is not considered the end of a line, and is included within the body of a record.

BINARY Store received files with variable length records and no record attributes. Records are written using the current "blocksize", The last record may be short, with its record size correctly indicated. The default "blocksize" for binary files is 510, so that a record together with its two-byte length field exactly fill a 512-byte VMS disk block. Any file which is just a stream of bytes can be stored as a BINARY file, and recovered intact later. This is the preferred file type for use in archiving non-VMS files. The longest possible record is 32765 plus two bytes for the RMS length field.

BLOCK Store received files exactly as they come in, byte for byte, with no formatting or record length information. When sending files, send the file data literally, including record attributes (if any), and ignoring all RMS attributes. Using a file type of BLOCK has proven effective when transferring files between the same application on unlike systems, for example Lotus 1-2-3 spreadsheets between VMS and MS-DOS. When receiving a file in this mode, any unused portions of the last block are filled with zeros.

FIXED Store the file as a fixed-length-record binary file. Any file received is stored as fixed length records with no record attributes, using the current "blocksize" (i.e. record length, 512 by default). Fixed-length 512-byte records is the format used for binary files such as VAX/VMS "EXE" files and RSX-11M/M+ "TSK" files. VMS BACKUP savesets are fixed-length-record files with a record-length ("blocksize") of 2048 or more. Since even the last record of the file is written with the whole record length (even if it is not filled), this format does not necessarily maintain the correct length of a file. It should normally only be used for fixed-length-record files coming from a VAX/VMS, PDP-11, or other system, when the fixed-length nature of the data must be preserved.

NAMING *keyword*

Determine the form of names to be sent with outgoing files and determine the translation performed on incoming file names. The choices are FULL, NORMAL_FORM and UNTRANSLATED.

FULL Kermit-32 will send full file names (including device, directory, file name, file type and version number). When receiving a file, Kermit-32 will perform no translation of the file name (which must therefore be a legal VAX/VMS file specification).

NORMAL_FORM

Kermit-32 will send only the file name and file type. When receiving a file, Kermit-32 will convert the file specification received to contain only uppercase letters, digits, and at most one period. Any other characters will be translated to "X". There will be at most 39 characters before the period (if any), and at most 39 characters afterwards. This forces the file name to be a valid VAX/VMS file specification for VMS versions 4.0 and later. This is the default style of file naming.

UNTRANSLATED
: Kermit-32 will send only the file name and file type. When receiving a file, Kermit-32 will not perform any conversions on the file specification, which therefore must be a legal VAX/VMS file specification. If you want to receive files with long names, use this option. To transfer files with VAX/VMS long names between two VMS 4.0-or-later systems, use this option on both sides.

SET HANDSHAKE

Syntax: `SET HANDSHAKE` *ooo*

Sets the half duplex line turnaround handshake character to the ASCII character whose octal value is *ooo*. Normally required for communication with half duplex systems like IBM mainframes in linemode.

SET IBM_MODE

Syntax: `SET IBM_MODE ON` *or* `OFF`

For communicating with IBM mainframes over half-duplex linemode connections. When IBM_MODE is set to ON, Kermit-32 will override the parity and local echo settings and use odd parity, local echo on, and also enable a handshake character of XON (control-Q, ASCII 021 octal). This feature allows Kermit-32 to exchange packets over half duplex connection with systems that wait for an XON before sending data.

The various features selected by this command can be overridden subsequently by SET PARITY, SET LOCAL_ECHO, and SET HANDSHAKE commands.

SET LINE

Syntax: `SET LINE [`*terminal-name*`]`

Specify the terminal name to use for file transfer or CONNECT; the *terminal-name* can be up to 255 characters long. If you issue this command using other than your job's controlling terminal, you will be running Kermit-32 in *local mode*, and you must log in to the remote system and run Kermit on that side in order to transfer a file. If you don't issue this command, Kermit-32 determines whether it is to run locally or *remotely* based on the default terminal line found when Kermit-32 is started. Kermit-32 uses a list of logical names to determine which terminal should be the default terminal line. The first of these names which translates to a terminal which is available (i.e., not allocated by some other process) is used. The logical names Kermit-32 tries are KER$COMM, SYS$INPUT, SYS$$OUTPUT, and SYS$COMMAND. If none of these translate to an available terminal, Kermit-32 is running *detached*, and a terminal must be specified by the SET LINE command before any actions can be performed. If a terminal is found, Kermit-32 is running locally if this is a terminal other than the one controlling the job (i.e., different from SYS$COMMAND), otherwise Kermit-32 is running remotely. You can also select the line directly in the CONNECT command; the command:

```
CONNECT TTA0
```

is equivalent to:

```
SET LINE TTA0
CONNECT
```

If you type SET LINE with no argument, you will deassign any previous assigned line and revert to remote mode on your job's controlling terminal.

SET SERVER_TIMEOUT

Syntax: SET SERVER_TIMEOUT *number*

This specifies the number of seconds between timeouts during server command wait, 0 specifies that no timeouts should occur during server command wait. When a Kermit server times out, it sends a NAK packet. Some systems cannot clear piled-up NAKs from their input buffers; if you're using such a system to communicate with a Kermit-32 server, and you expect to be leaving the server idle for long periods of time, you should use this command to turn off server command-wait timeouts. This command is also useful when a server is connected to a modem that is waiting for a call to come in, in which case the server's NAKs could confuse the modem's autodialer.

SET TRANSMIT

Syntax: SET TRANSMIT DELAY *integer*, SET TRANSMIT ECHO ON/OFF

It is possible to set a few parameters associated with the raw TRANSMIT command that vary both what the user sees on the screen as well as the speed of the transmit.

SET TRANSMIT DELAY
: This parameter is the amount of time to delay after each carriage return is transmitted. Valid delay values range between 0 (the default) and 9 tenths of a second. The format of the command is: SET TRANSMIT DELAY *d* Where *d* is a single decimal digit representing tenths of a second.

  Some remote hosts may not be able to receive the characters as fast as Kermit-32 can send them. The TRANSMIT DELAY can be used to slow up the transfer by adding a slight delay after each line is sent.

  The transfer also runs slower if the transmit echo is on, and the remote system is echoing the characters as it receives them. If the transmit delay is set to 9 tenths of a second, the remote system is echoing characters, the transmit echo is on, and the remote system still cannot keep up, then the connection should be made at a slower baud rate.

  Conversely, the file transfer speed can be increased by: setting the delay to 0 and the echo off, stopping the remote system from echoing the characters it receives, and connecting at higher baud rates.

SET TRANSMIT ECHO
: This command controls what the user sees on the screen during the file transfer. The format of the command is SET TRANSMIT ECHO ON or OFF. By default, the transmit echo is left off and the user sees the number of each line after it has been transmitted. With transmit echo on, the user sees whatever the remote system would normally echo back to him while he is typing in a file. Note that turning the echo on typically slows the file transfer down.

The SHOW Command

Syntax: `SHOW` [*option*]

The SHOW command displays various information:

ALL
: All parameters.

BLOCK_CHECK_TYPE
: The block check type being requested.

COMMUNICATIONS
: Parameters affecting the terminal line being used for communication.

DEBUGGING
: Debugging mode in effect, if any.

DELAY
: The number of seconds Kermit-32 will delay before starting a SEND or RECEIVE command when in remote mode.

ESCAPE The current escape character for the CONNECT processing.

FILE_PARAMETERS
File blocksize, type, file naming, and incomplete file disposition.

INCOMPLETE_FILE_DISPOSITION
The action to take when a transfer is aborted.

LINE Terminal line in use.

LOCAL_ECHO Whether characters should be echoed locally when CONNECTed.

PACKET For incoming and outbound packets.

PARITY The parity type in use.

RECEIVE For inbound packets.

RETRY The number of retries to be done on bad packets.

SEND For outbound packets.

TRANSMIT Parameters for TRANSMIT command.

VERSION The program version number of Kermit-32.

1.4.6. Program Management Commands

The HELP Command

Syntax: `HELP [`*topic* `{`*subtopic*`}]`

Typing HELP alone prints a brief summary of Kermit-32 and its commands. You can also type

`HELP` *`command`*

for any Kermit-32 command, e.g. "help send" or "help set parity" to get more detailed information about a specific command. The HELP feature depends on the Kermit-32 help file being correctly installed on your system.

The EXIT and QUIT Commands

Syntax: `EXIT`

Exit from Kermit-32. You can also exit from the Kermit-32 when it is waiting for a command by typing a control-Z. When Kermit-32 is running remotely, two control-Y's will abort the transfer, bringing Kermit-32 back to command mode. The two control-Y's must be typed together; if a timeout occurs between them the first is ignored. When Kermit-32 is running locally, two control-Y's will stop Kermit-32 and return you to DCL. You will be able to CONTINUE if you do not perform any command which runs a program. However, after continuing, control-A, control-X and control-Z will no longer be accepted as commands.

`QUIT` is a synonym for EXIT.

The LOG Command

Syntax: `LOG` [*option* [*filespec*]]

Log the specified option to the specified file:

SESSION	During CONNECT log all characters that appear on the screen to the specified file. During CONNECT, the session log can be temporarily turned off during the remote session by typing the escape character followed by Q (for Quit logging), and turned on again by typing the escape character followed by R (for Resume logging).
TRANSACTIONS	During file transfer, log the progress of each file. Transaction logging is recommended for long or unattended file transfers, so that you don't have to watch the screen. The log may be inspected after the transfer is complete to see what files were transferred and what errors may have occurred.
DEBUGGING	Log debugging info to the specified file. If no SET DEBUGGING command was previously issued, the file will be opened and no information written. If DEBUGGING is turned on (either via the SET DEBUGGING command or by typing control-D during a local transfer), the packet debugging information will be written to the file. Packet format is described in *Kermit, A File Transfer Protocol*, Frank da Cruz, Digital Press (1987).

Any log files are closed when you EXIT or QUIT from Kermit. You may explicitly close a log file and terminate logging by using the LOG command without a file specification.

The STATUS Command

Syntax: `STATUS`

The current status of Kermit-32 will be displayed. This includes the number of characters that have been sent and received from the remote Kermit. Also included is an estimate of the effective baud rate of the transfer. This number is not intended to be exact, but only an indication of what range of throughput has been provided.

1.5. Raw Upload and Download

The TRANSMIT Command

Syntax: `TRANSMIT` *file-spec*

The TRANSMIT command allows you to upload files "raw" to systems that don't have a Kermit program available. Note that there is no error checking or packets involved in this method of file transfer.

This command does a raw transmit of an ASCII file, one character at a time, with carriage returns (no line-feeds) at the end of each line. It is used with Kermit-32 in local mode. The user must first prepare the remote host to receive the file by starting an edit session in input mode. Then the user can escape back to Kermit-32 and issue the TRANSMIT command. After the transmit is finished, the user then CONNECTs back to the remote host again and ends the edit session.

During a file transmit, the following control characters can be used to affect the transfer in progress:

CTRL-C	Cancel the transmit
CTRL-X	Cancel the file currently being transmitted
CTRL-Z	Cancel the file group currently being transmitted

See SET TRANSMIT for information about controlling echo and delays.

The LOG SESSION Command

Syntax: `LOG SESSION` *file-spec*

"Raw Download" is the term commonly used to describe the capture of a remote file on the local system, without any kind of error detection or correction. This allows you to obtain files from remote systems that do not have Kermit, but with the risk of loss or corruption of data.

Kermit-32 provides raw downloading via the LOG SESSION command during CONNECT to a remote system. The session log is described above. To use session logging to capture a file:

1. Run Kermit on the VAX/VMS system.

2. SET LINE to the terminal line through which you will be connected to the remote system.

3. Perform any required SET commands to condition Kermit for communication with the remote system.

4. CONNECT to the remote system and log in.

5. Condition your job on the remote system not to pause at the end of a screenful of text, and give whatever commands may be necessary to achieve a "clean" terminal listing -- for instance, disable messages from the system or other users.

6. Type the appropriate command to have the desired file displayed at the terminal, *but do not type the terminating carriage return*. On most systems, the command would be "type", on Unix it's "cat".

7. Escape back to Kermit-32 and give the LOG SESSION command with the file specification where you wish to store the data.

8. CONNECT back to the remote system and type a carriage return. The file will be displayed on your screen and recorded in the session log file.

9. Escape back to Kermit-32 and give the LOG SESSION command without a file specification to close the session log file.

The file you specified will contain everything that was typed on your screen. You will probably find that some editing necessary to remove extraneous prompts, messages, padding characters, or terminal escape sequences, or to fill in lost or garbled characters.

Use the TRANSMIT command for raw uploading.

1.6. Installation of Kermit-32

VMS Kermit-32 comes in 3 forms: Hex, Macro source, and Bliss source. Each can be used as the basis for installation.

Before beginning, make a special directory for VMS Kermit and read the files `VMS*.*` from the Kermit distribution tape into this directory. Columbia's 9-track Kermit tapes are written with blocksize 8192, which is 4 times larger than the default tape blocksize for VMS. You should mount these tapes on the VMS system with the following command:

```
MOUNT/BLOCK=8192/DENSITY=1600 MTA0: KERMIT
```

(or substitute some other tape drive name for `MTA0:`) Do not use the `/FOREIGN` switch. Once the tape is mounted, you can use normal VMS COPY commands to copy the files from the tape. For instance, if you have defined your Kermit directory to have logical name KER:, you can use the following command to copy the VMS Kermit files into this directory:

```
$ copy mta0:vms*.* ker:
```

You might also have received Kermit on a TK50 tape cartridge that contains a VMS BACKUP saveset, in which case do this to get the files off:

1. SET DEFAULT to the directory under which you want the various Kermit subdirectories created.

2. Physically mount the TK50 cartridge, and type "MOUNT $TAPE1/FOREIGN".

3. Type "BACKUP/LOG $TAPE1/SAVE [.*]".

Installation Procedure

If you are running a pre-4.0 version of VMS, ignore the following material and skip ahead to the section **Kermit-32 for Old VMS Versions**.

At present, there is no VMSINSTAL "kit" for Kermit-32. However, there is a DCL procedure that will do most of the installation work for you. It is called VMSINS.COM. To run it, type:

```
$ @vmsins
```

It will ask you the question "Rebuild from sources? (YES or NO)". If you reply NO, then the Kermit task image will be decoded from the the VMSMIT.HEX file into KERMIT.EXE. If you reply YES, you will be given the choice of building the program from the Macro-32 sources (which are generated by the Bliss compiler from the original Bliss source code) or from the Bliss itself. All sites can build from Macro, but only those sites with Bliss compilers can build from the Bliss.

After building the KERMIT.EXE file, the VMSINS procedure copies it into SYS$SYSTEM, and then builds and installs the Kermit-32 help file in the system-wide help library (SYS$HELP:HELPLIB.HLB) so that users can get help for Kermit by typing "help kermit" at the DCL prompt, and it will also build and install SYS$HELP:KERMIT.HLP so that the HELP command will work from within Kermit.

HEX, Macro, or Bliss?

The VMSMIT.HEX file is built from KERMIT.EXE under the oldest version of VMS that the developers have access to (for example VMS 4.5). If you are running that version of VMS or later, then you should reply NO to the "Rebuild from sources? (YES or NO)" question.

If you are running an older version of VMS than the one under which the Kermit that forms the basis of the hex file was linked, then you will not be able to run it on your VMS system, because of a runtime library conflict. In that case, you should reply YES to the question, and VMSINS will try to build the program from the Macro-32 assembly language source code using your system's MACRO command. This should build a working version of Kermit-32 on all VAX/VMS systems 4.0 or later.

The only reason for building from the Bliss source is if you have made changes to Kermit-32. It is recommended that you only work on the Bliss source, and not the Macro source. If you make changes to the macro source, there is no way to carry them forward to the Bliss code, which is the true source code for the program. If you do intend to make changes to the Bliss code, be sure to contact Columbia University's Kermit Distribution Center first to make sure you are working from the latest release and that nobody else has already done, or is working on, the same thing.

Kermit-32 for Old VMS Versions

If you are running a pre-4.0 version of VAX/VMS, then you will have to install a very old version (3.1) of Kermit-32, rather than the current version, until you upgrade your VMS version. To use version 3.1 of VMS Kermit:

1. Rename `VMSMIT.HEX` to `VMSV33.HEX`
2. Rename `VMSV31.HEX` to `VMSMIT.HEX`
3. Run the VMSINS procedure and reply NO to the "Rebuild from source" question.

Note that the help files which are installed apply to the current release, `3.3.126`, rather than to version 3.3.

Defining a Kermit Command

You should define a system-wide symbol for Kermit as a "foreign command", for example in your `SYS$MANAGER:SYLOGIN.COM` (system-wide login command) file, like this:

```
KERMIT :== $SYS$SYSTEM:KERMIT.EXE
```

so that users can run Kermit just by typing its name.

Files

Kermit-32 is built from a number of BLISS-32 sources and one MACRO-32 source. In order to make it possible for sites without BLISS-32 to build, MACRO-32 sources generated by BLISS-32 are also included for all of the BLISS modules. The following files are distributed as part of Kermit-32:

`VMSTT.BLI` Common BLISS source for the terminal text output support.

`VMSGLB.BLI` Common BLISS source for the global storage for VMSMSG.BLI.

`VMSMSG.BLI` Common BLISS source for the protocol handling module.

`VMSCOM.REQ` Common BLISS require file which defines various common parameters. This is required by VMSMSG.BLI. This file must be renamed to KERCOM.REQ.

`VMSMIT.BWR` "Beware File" for Kermit-32 (read it!).

`VMSMIT.BLI` BLISS-32 source for the command parser, and some basic support routines.

`VMSFIL.BLI` BLISS-32 source for the file I/O.

`VMSTRM.BLI` BLISS-32 source for the terminal processing. This handles the driving of the terminal line for the transfers and the connect command processing.

`VMSSYS.BLI` System interface routines for the Kermit generic command processing.

`VMSGEN.MAR` Macro-32 source file that contains the REMOTE command text that is given to VMS. Sites desiring to change what DCL commands are used to process the various generic server commands can make those changes in this source. This also contains the text of the help message returned in response to the server generic help command.

`VMSERR.MSG` MESSAGE source for error messages used by VAX/VMS Kermit.

`VMSERR.REQ` BLISS-32 require file which defines the error codes. This is REQUIREd by the BLISS-32 sources.

`VMSMIT.MSS` SCRIBE source file for VMSMIT.DOC (this document).

`VMSMIT.RNH` RUNOFF source for the help files for VAX/VMS Kermit. When this is run through RUNOFF with /VARIANT=SYSTEM, it produces a .HLP (VMSSYS.HLP) file suitable for inserting into the system help library (SYS$HELP:HELPLIB.HLB) to provide a KERMIT topic for the system HELP command. When run through RUNOFF without the /VARIANT=SYSTEM, it produces a .HLP file (VMSUSR.HLP) to be stored on SYS$HELP: for use by the Kermit HELP command.

`VMSSYS.HLP` RUNOFF output file for system wide Kermit HELP.

`VMSUSR.HLP` RUNOFF output file for Kermit's HELP command.

`VMSREN.COM` Command file to rename VMS*.* to KER*.*.

`VMSINS.COM` Command file to build and install VAX/VMS Kermit.

`VMSMIT.HEX` A hexified version of .EXE file for VMS Kermit. This file can be dehexified using the supplied program. In the hexified form, the file should be transferable over any medium which handles normal text. This is the most reliable copy of the executable version of VMS Kermit.

`VMSHEX.MAR` Source for the hexification program. This is the program which was used to produce VMSMIT.HEX. It can also be used to produce hexified version of any (or at least almost any) Files-11 file. The dehexification program should then be able to reproduce a copy of the original file with the file parameters correctly set. Note that the format used for the hexified files is basically Intel hex format. There are some additional records used to store the record format, etc. Also, the file name as typed to the prompt from VMSHEX is stored in the hexified version of the file for use by the dehexification program. By doing this, it is possible to store more than one binary file with a single hexified file.

`VMSDEH.MAR` Source for the dehexification program.

`VMSV31.*` Version VMS Kermit, the last version that will run under release 3.x of VMS. Versions 3.2 and later require VMS release 4.0 or later.

`VMSV3x.MEM` ocumentation on the changes between releases 3.1 and 3.1, and 3.2 and 3.3 of Kermit-32, and additional installation information.

OTHER INSTALLATION CONSIDERATIONS

As distributed, Kermit-32 should work on any VAX/VMS system (version 4.0 and later). Customization is possible with or without a BLISS-32 compiler. Default parameter values may be changed by changing the appropriate LITERALs in the BLISS-32 source for VMSMSG, or the actual values which are stored in the routine MSG_INIT in the MACRO-32 source for VMSMSG.

Sites can also easily change the commands which are used for processing the generic server functions (REMOTE commands when running as a server). The text which makes up these commands is in the file VMSGEN.MAR, along with the text of the REMOTE HELP message. This allows a site to make use of local programs for performing some of the commands (perhaps using FINGER to perform the WHO command, etc.).

If you want to allow your users to assign external terminal lines for connecting to remote systems from the VAX, e.g. by dialing out, you will have to configure those lines to allow the desired access. Otherwise, users will get a message like "No privilege for attempted operation" when they do a SET LINE command. Sample commands for terminal TXA0: might include:

```
$ SET PROTECTION=(W:R) TXA0:/DEVICE
```

or

```
$ SET PROTECTION=(W:RWLP)/DEVICE/OWNER=[1,4] TXA0:
```

or

```
$ SET ACL/OBJECT=DEVICE/ACL=(IDENTIFIER=INTERACTIVE,OPTIONS=NONE,-
  ACCESS=READ+WRITE) TXA0:
```

Consult your VAX/VMS system manager's manual for the ramifications (especially on security) of each of these commands.

Index

Table of Contents