UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 79

(August 1985)

Title:

KERMIT on EMAS

Author: Adam Albert-Recht

Contact: Advisory service

Software Support Category: n/a

Synopsis

This User Note describes how to use the implementation of KERMIT on EMAS produced by the Edinburgh Regional Computing Centre. It is intended to provide enough information for a novice Kermit user to be able to transfer data between EMAS and another Kermit system on a microcomputer or another mainframe.

The information in this edition applies to version 2.5 and later releases.

Keywords

File transfer, KERMIT

Edinburgh Regional Computing Centre

James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ. Telephone 031-667 1081

CONTENTS

1: INTRODUCTION

This user guide describes the Kermit implementation on EMAS produced by the Edinburgh Regional Computing Centre. It is intended to provide enough information for a novice Kermit user to be able to transfer data to and from EMAS to another Kermit system. Other Kermit systems are described only in passing: thus a user would almost certainly need to consult the equivalent user guide for the Kermit system on the other machine. The examples in this document assume a transfer between a microcomputer and EMAS. The EMAS end of the transfer is always referred to as "EMAS Kermit", while the other end is referred to severally as "local Kermit", "micro Kermit" or just "other Kermit".

The guide is divided into several chapters. Chapter 2 is a general overview of Kermit as a whole, and explains its advantages as a file transfer system over "dumb capture" programs. The next chapter describes the command language that EMAS Kermit uses. Following that are chapters that describe how to use EMAS Kermit to transfer data.

The final chapters comprise the "reference section". They describe in full detail the commands available in EMAS Kermit, grouping them by functionality (i.e. "Commands for file transfer" etc).

2: AN OVERVIEW OF KERMIT

Kermit is a system, devised at the Center for Computing Activities at the University of Columbia, New York (CUCCA), to permit the simple and flexible transfer of data from a microcomputer to a mainframe or another microcomputer. CUCCA retain the copyright on Kermit, but have published full information on it and permit anyone to implement it on their own machines, provided this is not done for commercial purposes. The result is that Kermit is now available on a very wide range of machines indeed: very few micros and mainframes now do not have a Kermit of some sort available for them.

The primary design aim of Kermit is to permit the transfer of any data whatsoever between systems, and to make the data usable on the system that receives it if this is possible. To illustrate why this is important, and not possible with simple systems, we can consider an ordinary terminal emulation system that allows data to be captured into files or sent from them.

Simple terminal emulator systems, such as those commercially available for, say, the BBC micro, would permit you to transfer files from EMAS in a rudimentary way. You would tell the emulator to copy any characters that appear on the screen into a file, then ask EMAS to list the file. The reverse process would let you input data into an EMAS file from your BBC discs.

The problems arise in the nature of the communications system that connects the micro to the mainframe, and how the mainframe itself uses this system. A character of data in a file occupies one byte, which consists of 8 binary digits or "bits". If you regard the pattern of bits representing a character as a number, this allows numbers ranging from 0 to 255 to be used. However, many communications systems will allow only seven of the eight bits to be transmitted along them. The most significant bit, termed the "parity bit", is used by the communications system as an error-checking device. Thus, even though you send a byte of 8 bits to the mainframe, it may receive only 7 of them. This immediately restricts the range of characters that can be sent to those whose codes are in the range 0 to 127.

A further restriction may be imposed if the communications system uses some of those characters for its own control purposes: thus systems often will use the

characters whose codes are 17 and 19 to prevent overloads occurring. In such systems, you cannot transmit these characters at all. To make matters even worse, some machines will (apparently arbitrarily) decide that you could not possibly want to send some characters, so, if you do send them, it will change them into something else entirely.

One method of file transfer which is extensively used in the Edinburgh environment is X-Talk. It avoids those of the above problems which occur on the ERCC network, but it is parochial, and there is a strictly limited range of machines on which it is implemented. Section 3 of this guide gives a rough comparison of X-Talk and Kermit, and suggests when you might use one or the other.

Kermit overcomes all these difficulties by encoding the data it sends according to a standard set of rules or "protocol". Kermit recognises that many characters cannot be transmitted down a communications line, so if those characters occur, they are translated into something that will be transmitted. The receiving end, of course, translates them back again to what they were. This technique guarantees that the data you send is the data that arrives, since Kermit uses special methods for detecting garbling and will repeat any transmissions that did not get through correctly.

Besides the problems of actually transferring data correctly, there is the problem of making it usable on the other end of the transmission link. If you are sending, say, a machine code program from a micro to EMAS, this is not a problem, since EMAS probably cannot understand the micro's machine code anyway. Nor does it matter if you use the EMAS system only as an archive: it is irrelevant how the data is held on EMAS, as long as when it is brought back to the micro it looks the same as when it was sent.

The usability problem does appear, though, if you want to move a file from a micro to EMAS and then actually use it on EMAS. You might, for instance, word process a file on the micro, then send it to EMAS to be printed. In this case, you may not want to transfer the data byte-for-byte, since the way the micro and EMAS denote things like the end of each line of text will almost certainly be different. What you require is that the file of printable lines on the micro, which you can process on that machine, becomes a file of printable text on EMAS, that can be processed there.

With Kermit the problem can easily be circumvented. The Kermit protocol defines a standard way of indicating the end of a printable line. When you send a file from the micro, Kermit will translate whatever ends the lines of text in your file into this standard form before sending the data. EMAS Kermit, seeing this standard end-of-line indicator, will translate it into its end-of-line character. You thus end up with a usable file of lines, with no extra characters anywhere.

The requirements you must meet before using Kermit are simple. You will need a version of Kermit in your micro, EMAS Kermit to be set up in your filespace, and a way of linking the machines, be it a network, an ordinary cable, or a piece of wet string. (This is a joke. Wet string won't work.)

3: X-TALK versus KERMIT

Transmission Errors

Kermit detects and corrects transmission errors during transmission. X-Talk detects and reports transmission errors at the end of a file transfer. The file must be retransmitted by the user.

Availability

Kermit is available for a wide range of micros and hosts (see Section 7). X-Talk is available for only a limited range of machines.

Micro-to-micro communications

All micro Kermits can be used for micro-to-micro file transfers, while X-Talk, being an asymmetric protocol, requires one micro to pretend to be a host. The "host" end of the protocol is only implemented on UCSD p-system micros.

Speed and cost

Between a micro and EMAS, Kermit is much slower and more expensive than X-Talk. However, between a micro and a VAX/VMS system, Kermit is very much less demanding on VAX CPU time. Moreover, under VMS version 4 and above, Kermit will work across the ERCC network to a VAX, which X-Talk will currently not do. See Appendix I for details of comparative costs and speeds.

Ease of Use

Kermit is not at all easy to use (as may be evident from this guide). X-Talk, however, is much more user-friendly.

Which should I use?

If you only want to transfer data to and from EMAS from a micro for which X-Talk is available, you should use X-Talk.
If you have a communications line susceptible to noise (for example a dial-up line), or you wish to transfer data to a VAX, or you wish to use a micro for which no X-Talk is available, then use Kermit.

4: USING EMAS KERMIT

In this section we shall look at how you start and stop EMAS Kermit, and also how its command language operates.

4.1 ENTERING EMAS KERMIT

To access the Kermit program on EMAS, type the once-only command:

```
Command:OPTION SEARCHDIR=MICROS.KERMDIR
```

Thereafter, to run Kermit, type:

```
Command:KERMIT
```

4.2 LEAVING EMAS KERMIT

To leave EMAS Kermit, you would normally use the EXIT command. However, if you wish to log out immediately, you can use the EMAS QUIT command.

4.3 EMAS KERMIT COMMAND LANGUAGE

You control what you want EMAS Kermit to do, and how it should do it, by giving it commands in its "command language". The format of the command language closely follows that used on most other Kermit implementations on other machines.

4.3.1 Command format

When you enter Kermit, you will see a prompt on the screen

```
Kermit-EMAS>
```

This indicates that EMAS Kermit is expecting you to type a command. You can type either a Kermit command, or an EMAS command preceded by "EMAS" (e.g. "EMAS FILES"). Like EMAS commands, Kermit commands can be typed in upper case, lower case, or any mixture of the two as you please.

Kermit's own commands all take the form of a command name, such as "SET", sometimes followed by one or more further pieces of information or "parameters", which must be separated from each other by spaces. For example, one command you might use is:

```
SET FILE-TYPE ASCII
```

to select text file transfer. Here the command is "SET", while "FILE_TYPE" and "ASCII" are parameters to the command.

The reference section at the end of this guide gives the complete specification of all the EMAS Kermit commands. The commands are grouped according to their function (such as "file transfer control"), rather than in one alphabetical list. Thus you will find variations of the SET command appearing in many places, grouped with other commands that function in the same area.

The reference section presents each command in a formal way to show you exactly what you are allowed to type at any point. You might find one command described as having a format:

```
SET BINARY-QUOTE <character>
```

for example. Here the fixed parts, that you would always include, are shown in capitals. The third part of the command, "<character>", is enclosed in "<>" brackets to show that here you must provide your own parameter value. The fact that it says "character" indicates that what you should type is a character of some sort: the description of the command will tell you what sort of character you must provide and why. Note that you do not type the "<>" brackets yourself here. Thus, a valid way of using this command might be:

```
SET BINARY-QUOTE %
```

where you have supplied the character "%" where it was expected. Additionally, lines shown in underlined writing may be typed verbatim at the console. Unless otherwise shown, these lines are assumed to be in response to the EMAS Kermit command prompt.

Sometimes, not all the parameters of some commands need be typed. In these cases, Kermit will take a "default" value for the parameter you did not supply. The reference section will tell you which parameters can be omitted, and what values Kermit will assume if they are omitted.

4.3.2. The TAKE file facility

As an alternative to typing commands in on the keyboard, you can place them in a file on EMAS (using some sort of text editor) known as a "TAKE file", and tell EMAS Kermit to read the commands from there instead of from the keyboard. This is done with the TAKE command.

EMAS Kermit will read the file in as though the characters were coming from the keyboard, and will obey its contents as commands. You can include either Kermit commands or EMAS commands in the file, the only exception being another TAKE command, which is not allowed. Additionally, you can place comment lines in the file to describe what it does: any line that starts with an exclamation mark ("!") will be ignored by Kermit.

When the end of the TAKE file is reached, Kermit will close it and revert to reading input from the keyboard.

5: TRANSFERRING FILES WITH KERMIT

The primary use of EMAS Kermit is to transfer files between it and a microcomputer. The methods used will be substantially the same whatever the other system is, since any Kermit system should be able to communicate with any other. Though the general techniques will be the same, the exact commands used to control the other Kermit will vary from one system to another. You will need to consult the user guide for the other system to discover how it should be controlled. In this section we shall cover in detail how EMAS Kermit is controlled.

5.1 PRINCIPLES

Transferring files with Kermit involves several discrete steps. We shall consider here the most common case of transfer to and from a micro.

1. The micro Kermit is entered and set up for the transfer. In particular, you may wish to tell Kermit what types of file are to be moved. You may also need to set any parameters for terminal emulation.

2. Terminal emulation mode is entered, and the local system is logged in to EMAS as though it were an ordinary terminal.

3. EMAS Kermit is started.

4. Commands can then be given to EMAS Kermit (from terminal emulation mode) and to the local Kermit (from local Kermit command mode). The two Kermit systems will communicate with each other using the standard Kermit protocol.

5. After the transfers are done, the terminal is logged out from EMAS.

In practice, the steps taken will range up and down this list as required. For example, EMAS Kermit parameters can be changed at any time, not only at the start, and if you are moving several types of file you will need to change them frequently. In the sections below we shall consider the various actions you will need to take: the order of doing them is up to you.

5.2 SETTING FILE TYPE

5.2.1. Binary files

These files contain data that is not primarily printable text, such as machine-code programs or word processor sources for your micro. When you transfer these files, you wish every byte in the file on the micro to appear unchanged in the file on EMAS, regardless of what it is.

You tell Kermit that you are handling binary files with the command:

```
SET FILE_TYPE BINARY
```

which tells it not to change any data that it either sends or receives. Note here that you may need to issue a comparable command to the local Kermit, to prevent it trying to manipulate the data. Some Kermits may not allow you to send pure binary data.

5.2.2. Printable text (ASCII) files

These files contain printable text. When you transfer one of these files, you do not necessarily want a byte-for-byte transfer, since the two machines may differ in how they store text files. The Kermit standards define a fixed way in which things such as end-of-line are transferred: EMAS Kermit will translate your data to this standard format, and the other end will then translate the standard format into whatever its own specific requirements are.

You tell EMAS Kermit that ASCII text files are to be transferred with the

```
SET FILE_TYPE ASCII
```

command.

5.2.3 Choosing the right file type

It is important to note that "binary" files mentioned here do NOT include executable objects on EMAS. All files read by and written by EMAS Kermit are of type CHARACTER (on EMAS), even if they contain an executable image for running on another machine. A good rule of thumb to use is that if the file is text, and you wish to be able to look at it on EMAS, then use an ASCII transfer. Otherwise, use BINARY mode. The default when Kermit is started is ASCII mode.

5.2.4 Sending eight bit data

A further point to consider when transferring files concerns whether you can transfer all the 8 bits in every byte. It is common for communications systems to mainframes to restrict data to only 7 bits in each byte: thus you can only normally send characters whose ASCII codes are in the range 0 to 127. However, some text files and every binary file will contain bytes from the whole character set, with codes from 0 to 255.

Kermit in general has a technique for overcoming this restriction, by encoding characters in the range 128 to 255 into special sequences that can be sent down any communications line. EMAS Kermit will use this technique automatically when appropriate. Almost all modern Kermits will use this technique, which is known as "eight bit prefixing", but you may encounter an older implementation on some machine that does not support it. In this case your data will be garbled in

transmission. There is, regrettably, no way round this problem from within Kermit.

5.3 GETTING STARTED

Once you have logged in, you can start the EMAS Kermit program.

EMAS Kermit is only able to operate as a normal Kermit (termed non-server mode). In this mode, you will need to give commands both to it and to the local Kermit for every file transfer (here a transfer of a group of files in one go counts as one operation), which will involve you in continual changes between local Kermit command mode and terminal mode.

WARNING: If you log onto EMAS through a TCP, you will need to issue the following command in response to the `Kermit-EMAS>` prompt:

```
Kermit-EMAS> set receive start_of_packet 2
```

and the following command in response to the Micro's Kermit command prompt:

```
Kermit-Micro> set send start_of_packet 2
```

as the TCP will attempt to intercept Kermit's default start-of-packet character which is ASCII 1 (SOH or Control+A). See section 6.3 for details.

5.3.1. Sending files to EMAS

To send a file to EMAS you use the command SEND on the local machine. You must also tell EMAS Kermit that a file is on its way.

a. In terminal mode, start EMAS Kermit, and issue the RECEIVE command. This tells it to expect a file from the local system. EMAS Kermit will then wait for something to happen.

b. Type the control sequence to return to the micro Kermit command mode. (See micro Kermit User Guide for details.)

c. Issue the micro Kermit SEND command.

Example:

```
Kermit-Micro> connect

[Connecting to host.  Type Ctrl+] C to return to micro]

Command: kermit

EMAS Kermit V2.3

Kermit-EMAS> receive doc

Ctrl+] C

[Back at Micro]

Kermit-Micro> send info.doc
```

5.3.2 Sending files from EMAS

Transferring files from a local machine to EMAS is the exact reverse of the above RECEIVE procedure: all you need to do is reverse the roles of the two machines.

a. In terminal mode, start the EMAS Kermit program, and issue its SEND command. This tells it to transfer a file to the local system. There will normally be a delay before anything happens - the interval may be anything from a few seconds upwards, and is intended to let you do the next step before the transfer starts.

b. Type the control sequence to return to the micro.

c. Issue the RECEIVE command to the local Kermit. When EMAS Kermit's delay time expires, it will start to send the file. The RECEIVE command tells the local Kermit to sit and wait until this happens.

Example:

```
Kermit-Micro> connect

[Connecting to host.  Type Ctrl+] C to return to micro]

Command: kermit

EMAS Kermit V2.3

Kermit-EMAS> send doc

Ctrl+] C

[Back at Micro]

Kermit-Micro> receive info.doc
```

5.4 HANDLING PROBLEMS

By design, Kermit is a highly reliable file transfer system, and performs considerably better than any "dumb capture" facility within a terminal emulator. The error-detection capabilities of Kermit ensure that data is transmitted correctly.

That said, there are some cases where you may need to abort a transfer.

The simplest way out of possible problems is for you to keep an eye on the progress of the transfer and see when it appears to be in trouble. The local Kermit will probably have some sort of display facility to show you how the transfer is going. If things seem to have ground to a halt, and the local Kermit is not able to do anything, you can abort the transfer at the EMAS end by typing the sequence of characters necessary to generate the `Int:` prompt, (for example 'ESC' on a TCP, 'CTRL+P B' on a PAD) then typing ABORT followed by return. This will not be echoed on the screen, but should return you to the Kermit prompt in a few seconds.

5.5 Protocol control

The rules by which files are transferred between Kermit systems are termed the "Kermit protocol". These rules define in detail how data should be transferred:

they specify how much can be sent in one chunk or packet, what control sequences indicate the start and end of a packet, what character encoding is to be used, and so on. In almost every case you will have no need to change any of these settings, since they are carefully chosen so that any Kermit can communicate with any other Kermit in just about every circumstance.

However, it is possible that you may come across cases where you need to change some of the protocol values, either to improve the performance of the file transfer mechanism, or because the standard settings are inappropriate and do not work.

The protocol values are changed by the SET command, and EMAS Kermit allows you to change all the possible values. The reference section details all the SET commands concerned and their effects. A detailed discussion of the various possibilities is beyond the scope of this user guide, since some understanding of the Kermit protocol is needed. You will find this protocol explained in the "Kermit Protocol Manual" (use issue 5 or later).

6: COMMANDS FOR THE CONTROL OF KERMIT

In this section, we shall look at the commands you can you can use to control the operation of EMAS Kermit.

6.1 General Commands

EXIT

This command causes EMAS Kermit to return to the `Command:` prompt at the end of a session.

The command has no parameters.

Example:

```
EXIT
```

HELP

This command causes on-line help information to be displayed using the EMAS VIEW system. On exit from this, you will be returned to the Kermit command prompt, where you left off.

The command has no parameters.

Example:

```
HELP
```

SHOW

This command displays the values of all the Kermit control values.

The command format is:

```
SHOW <value>
```

The parameter is:

<value> This specifies the name of the control value to be shown.

Example:

```
SHOW PROMPT
```

causes the current prompt string (Kermit-EMAS> by default) to be displayed. Additionally,

```
SHOW ALL
```

displays all the control values.

EMAS

This command passes its parameters to the EMAS command line interpreter to be executed as a command.

The command format is:

```
EMAS <EMAS command>
```

The parameter is:

<EMAS command> This is an EMAS command to be executed.

Example:

```
EMAS FILES
```

causes the FILES command on EMAS to be executed, as if typed directly at the Command: prompt.

SET PROMPT

This command causes the Kermit command prompt to be changed.

The command format is:

```
SET PROMPT <string>
```

The parameter is:

<string> This specifies the Kermit command prompt string.

By default, the Kermit command prompt is Kermit-EMAS>.

Example:

```
SET PROMPT EMAS:
```

causes the command prompt to subsequently be set to EMAS:.

SET DELAY

This command changes the time that EMAS Kermit delays before sending its first packet, i.e. the amount of time required by you to escape back to the micro and initiate the RECEIVE... end of the transfer.

The command format is:

```
SET DELAY <number>
```

The parameter is:

<number> This specifies the time, in seconds, that EMAS Kermit will wait before sending its first packet.

By default, the delay is 5 seconds.

Example:

```
SET DELAY 10
```

causes the delay prior to sending the first packet to be set to 10 seconds.

TAKE

This command causes EMAS Kermit to read all further commands from a file instead of from the keyboard.

The command format is:

```
TAKE <filename>
```

The parameter is:

<filename> This specifies the name of a file containing EMAS Kermit commands, in the same format as if they were typed at the keyboard.

Once you issue the TAKE command, EMAS Kermit will read characters from the specified file instead of the keyboard. Any Kermit or EMAS command can be issued from within a TAKE file, except a further TAKE command.

6.2 Commands for file transfer

In this section we shall look at the detailed format of the commands that you use to transfer files using EMAS Kermit, and to control how EMAS Kermit will perform the transfers.

RECEIVE

This command causes EMAS Kermit to wait for a file transfer to be started by the local system. You will thus need to issue a SEND command to the other Kermit in order to make something happen.

The command format is:

```
RECEIVE <EMAS-file>
```

The parameter is:

<EMAS-file> This parameter is optional, and specifies the name of a file on EMAS into which you wish data to be transferred. It must be a legal EMAS filename.

When the command is issued, EMAS Kermit will wait passively for a signal from the other Kermit that a file transfer is beginning. This signal will include the name of the file that is being sent: if you have included the <EMAS-file> parameter, this name is for information only, and the data will be written to the file you have identified.

If you omit the <EMAS-file> parameter, EMAS Kermit will attempt to generate a suitable EMAS filename from the name supplied by the remote system.

If the filename is the same as a file that already exists, then by default EMAS Kermit will try to alter it until there is no clash. However, you can turn off this protection using the SET FILE_PROTECTION OFF command.

SEND

This command sends a file to the local Kermit.

The command format is:

```
SEND <EMAS-file> {, <EMAS-file> }
```

The parameters are as follows:

<EMAS-file> This parameter is mandatory. It specifies the name of the file on EMAS that you want to send, and can be any legal EMAS text file.

You must then escape back to the local machine, and issue a RECEIVE command there to prepare it for this SEND.

You may send a group of files simply by specifying a list of filenames, separated by commas.

SET FILE_PROTECTION

This command tells EMAS whether it should create a unique filename for files being RECEIVEd, or just let any already existing file be overwritten.

The command format is:

```
SET FILE_PROTECTION <on/off>
```

The parameter is:

<on/off> ON or OFF. ON means that a unique filename will be generated if necessary.

By default, EMAS Kermit will generate a unique filename in the case of a clash.

Example:

```
SET FILE_PROTECTION OFF
```

causes EMAS Kermit not to bother about possibly overwriting a file when receiving.

SET FILE_TYPE ASCII

This command tells EMAS Kermit that files it transmits and receives are to be taken as containing printable ASCII text and to transform them accordingly.

The command format is:

```
SET FILE_TYPE ASCII
```

After using this command, EMAS Kermit treats all files as containing printable ASCII text, with the end of each line indicated by the EMAS default end-of-line character. When sending a file, it will transform every occurrence of this character into Kermit's standard end-of-line indicator, and the remote Kermit should then change this into whatever the standard representation of end-of-line is on its own system. When receiving files, EMAS Kermit will change every occurrence of the standard Kermit indicator into the standard EMAS end-of-line character.

You may need to give an equivalent command to the other Kermit system to make it treat the data correctly.

By default, EMAS Kermit treats files as ASCII, with end of line indicated by an LF byte.

SET FILE_TYPE BINARY

This command tells EMAS Kermit that all files it receives or sends should be treated as containing binary data.

The command format is:

```
SET FILE_TYPE BINARY
```

After using this command, EMAS Kermit will transmit the bytes from a file exactly as they are, and will not change any of them. Similarly, the data it receives will be written to a file with no alteration.

You may need to give an equivalent command to the other Kermit system to make it treat the data correctly.

By default, EMAS Kermit treats files as ASCII text.

Example:

```
SET FILE_TYPE BINARY
```

makes EMAS Kermit treat files as containing binary data.

6.3 Commands for detailed protocol control

The commands described in this section are used to exert detailed control over the Kermit protocol. As explained earlier, it is unlikely that you would ever need to use these commands, unless you log on to EMAS through a TCP. You should consult the Kermit Protocol Manual for a detailed description of the facilities they

control.

SET BINARY_QUOTE

This command defines the eight-bit prefix character that EMAS Kermit will ask the other Kermit to agree to use.

The command format is:

```
SET BINARY_QUOTE <character>
```

The parameter is:

<character> The printable character to be used.

By default, EMAS Kermit will attempt to use the "&" character as its eight-bit prefix.

Example:

```
SET BINARY_QUOTE %
```

sets the eight-bit prefix character to be "%".

SET REPEAT_QUOTE

This command defines the repeat count prefixing character that EMAS Kermit will ask the remote system to agree to use.

The command format is:

```
SET REPEAT_QUOTE <character>
```

The parameter is:

<character> The printable character to be used.

By default, EMAS Kermit will attempt to use the "~" character as its repeat count prefix.

Example:

```
SET REPEAT_QUOTE ^
```

sets the eight-bit prefix character to be "^".

SET RECEIVE PACKET_LENGTH

This command defines the maximum packet size that EMAS Kermit will ask the remote system to send.

The command format is:

```
SET RECEIVE PACKET_LENGTH <number>
```

The parameter is:

<number> The maximum size of packet wanted from the remote system, in the range 35 to 92.

By default, EMAS Kermit asks the remote system to use a maximum packet size of 94 bytes.

Example:

```
SET RECEIVE PACKET_LENGTH 60
```

causes EMAS Kermit to request the remote system to use a maximum packet size of 60 bytes.

SET RECEIVE START_OF_PACKET

This command defines to EMAS Kermit the character that the remote system will use to indicate the start of each packet.

The command format is:

```
SET RECEIVE START_OF_PACKET <number>
```

The parameter is:

<number> The numeric code of the character the remote system will use, in the range 0 to 31.

By default, EMAS Kermit will expect the remote system to precede each packet with the ASCII 1 (SOH or Control+A) character. However, if your connection is through a TCP this character cannot be used, so you must use this command to change the expected value to something else.

Example:

```
SET RECEIVE START_OF_PACKET 2
```

tells EMAS Kermit that the remote system will precede each packet with the ASCII 2 (STX or Control+B) character. Note that having done this, you will need to change the SEND START_OF_PACKET character at the local Kermit to the same thing.

SET RECEIVE RETRIES

This command defines the number of times that EMAS Kermit will attempt to receive any particular packet before giving up.

The command format is:

```
SET RECEIVE RETRIES <number>
```

The parameter is:

<number> The maximum allowable number of retries before failure is acknowledged.

By default, EMAS Kermit will retry 10 times before admitting defeat.

Example:

```
SET RECEIVE RETRIES 20
```

sets the maximum number of retries to 20.

SET SEND END_OF_LINE

This command defines the value that EMAS Kermit uses to signal the end of a packet.

The command format is:

```
SET SEND END_OF_LINE <number>
```

The parameter is:

<number> The number of the control character in the range 0 to 31.

By default, EMAS Kermit terminates its packets with Carriage Return (ASCII 13).

Example:

```
SET SEND END-OF-LINE 10
```

causes EMAS Kermit to terminate its packets with Line Feed (ASCII 10).

SET SEND RETRIES

This command defines the number of times that EMAS Kermit will attempt to send any particular packet before giving up.

The command format is:

```
SET SEND RETRIES <number>
```

The parameter is:

<number> The maximum allowable number of retries before failure is acknowledged.

By default, EMAS Kermit will retry 10 times before admitting defeat.

Example:

```
SET SEND RETRIES 20
```

sets the maximum number of retries to 20.

SET SEND PADCHAR

This command defines the padding character with which EMAS Kermit will precede the first packet to be sent in a transfer. Subsequent packets will be preceded by the character requested by the remote system in its SEND-INIT packet.

The command format is:

```
SET SEND PADCHAR <number>
```

The parameter is:

<number> The numeric code of the character to be used, in the range 0 to 255.

By default, EMAS Kermit uses a pad character of NUL (ASCII 0)

Example:

```
SET SEND PADCHAR 3
```

sets the pad character used to be ASCII 3.

SET SEND PADDING

This command defines the number of pad characters EMAS Kermit will send before the first packet in a transfer. Subsequent packets will be preceded by the number requested by the remote system in its SEND-INIT packet.

The command format is:

```
SET SEND PADDING <number>
```

The parameter is:

<number> The number of pad characters to be sent, in the range 0 to 255.

By default, EMAS Kermit sends no pad characters.

Example:

```
SET SEND PADDING 21
```

causes EMAS Kermit to send 21 pad characters before its first packet.

SET SEND QUOTE

This command defines the character that EMAS Kermit will use to prefix control characters in data packets that it sends.

The command format is:

```
SET SEND QUOTE <character>
```

The parameter is:

<character> The printable character to be used.

By default, EMAS Kermit uses the "#" character.

Example:

```
SET SEND QUOTE $
```

causes EMAS Kermit to prefix control characters it sends in data packets with a "$".

SET SEND START_OF_PACKET

This command defines the character that EMAS Kermit will send to indicate the start of every packet.

The command format is:

```
SET SEND START_OF_PACKET <number>
```

The parameter is:

<number> The numeric code of the character to be sent, in the range 0 to 31.

By default, EMAS Kermit sends the ASCII 1 (SOH or Control+A) character in front of every packet.

Example:

```
SET SEND START-OF-PACKET 5
```

causes EMAS Kermit to precede every packet it sends with the ASCII 5 (ENQ or Control+E) character.

7: KERMIT AVAILABILITY

Kermit is available for the following mainframe computers:

Machine	System	Language	Date	Version
Cyber 170	NOS	Fortran 77	07/09/84	2.2
DEC 20	TOPS 10	MACRO 20	15/11/84	4.2(253)
Data General	AOS	Ratfor	14/09/84	
Burr. B6800		Algol/NDL	15/02/85	
Burr. B7900		Algol	23/04/85	
Many	UNIX Sys 3	C	30/05/85	4C
PDP-11,..	UNIX V7	C	30/05/85	4C
DEC VAX	VMS	C	30/05/85	4C
DEC VAX	UNIX 4xBSD	C	30/05/85	4C
SUN,..	UNIX 4xBSD	C	30/05/85	4C
IBM 370	VM/CMS	IBM assembler	01/02/84	
Cray 1 & XMP	CTSS	Fortran 77	08/02/85	
Harris 800	VOS	Pascal, assembler	11/02/85	
H/Well CP6	IBEX	Pascal	04/04/85	
H/well DPS/8	GCOS	B	21/03/85	1.1
H/well DPS8, 66	GCOS3 or 8	C	05/10/84	3.0
HP3000	MPE	SPL	25/10/84	1.0
HP1000	RTE	Fortran	14/09/84	1.0
GEC 4000	OS4000	Babbage		
IBM 370	MUSIC	Assembler	26/12/84	1.0
DEC 10	TOPS 10	MACRO 10	01/06/84	3(124)

Pro 350	POS,RT11	MACRO-11	20/03/85	2.26
PDP-11	RSX11M,M+	MACRO-11	20/03/85	2.26
PDP-11	RSTS	MACRO-11	20/03/85	2.26
PDP-11	RT11,TSX+	MACRO-11	20/03/85	2.26
PDP 11	MUMPS/11	MUMPS	11/04/84	
IBM 370	MTS	Assembler, Pascal	06/01/84	
Honeywell	MULTICS	PL/1	20/09/84	2.0h
PRIME	PRIMOS	PL/P (PL/1)	10/02/84	
DG Nova	RDOS	Fortran	14/09/84	
HP3000	Software Tools	Ratfor	18/02/84	1n
Univac	Software Tools	Ratfor	11/06/84	1n
Tandem	Nonstop	TAL	30/11/84	0.0
IBM 370	MVS/TSO	Assembler	20/03/85	1.0
Univac 1100	EXEC	Assembler	08/10/84	
Univac 1100	NOSC	Pascal	08/10/84	2.0
VAX	VMS	Bliss32/Macro32	25/04/85	3.1.065

and for the following microcomputers:

Machine	**System**	**Language**	**Date**	**Version**
Luxor ABC800	ABCDOS	BASIC-II	08/05/84	2.2
Fujitsu M16	CP/M-86	ASM86	03/12/84	2.9
Tektronix 4170	CP/M-86	ASM86	03/12/84	2.9
NEC APC	CP/M-86	ASM86	03/12/84	2.9
Rainbow	CP/M-86	ASM86	03/12/84	2.9
Alpha M 68000	AMOSL	Alpha asm68k	11/02/85	1.0
Apple II	Apple DOS	Apple assembler	09/10/84	2.?
NEC APC binaries	CP/M-86	ASM86	03/12/84	2.9
Apollo	Aegis	Fortran	04/04/85	1.0
Apple II	Apple DOS	DEC-10 Cross	12/09/84	2.1A
ACT Apricot	MSDOS	MASM	14/11/84	1.20
Atari Home Comp.	DOS	Action!	09/01/84	
BBC Micro	OS 1.20	ADE assembler	10/06/85	1.02
Commodore 64	DOS	DEC 10 Cross	04/03/85	1.3
Commodore 64	DOS	Forth	08/02/85	1.5
TRS80 color	Ext.color BASIC	EDTASM	21/03/85	1.1
ATT 3Bx,..	UNIX Sys 5	C	30/05/85	4C
Macintosh		C(SUMACC)	30/05/85	0.8
DEC Pro-350	Venix V1	C	30/05/85	4C
IBM PC/AT,..	Xenix/286	C	30/05/85	4C
IBM PC/XT	PC/IX	C	30/05/85	4C
NCR Tower	OS1.02	C	30/05/85	4C
Apple II	CP/M-80	ASM	19/02/85	4.05
Aculab Z80B	CP/M-80	ASM	26/06/85	4.05
Bigbrd II	CP/M-80	ASM	19/02/85	4.05
CPT-85xx	CP/M-80	ASM	19/02/85	4.05
DEC VT180	CP/M-80	ASM	19/02/85	4.05
DECmate II	CP/M-80	ASM	19/02/85	4.05
Delphi 1000	CP/M-80	ASM	19/02/85	4.05
Cifer 1886	CP/M-80	ASM	28/05/85	4.03
Gen. 2.2	CP/M-80	ASM	19/02/85	4.05
Gen. 3.0	CP/M-80	ASM	19/02/85	4.05
H/Z-89	CP/M-80	ASM	19/02/85	4.05
H/Z-100	CP/M-80	ASM	19/02/85	4.05
Kaypro II	CP/M-80	ASM	19/02/85	4.05
Lobo Max80	CP/M-80	ASM	19/02/85	4.05
Morrow D.1	CP/M-80	ASM	19/02/85	4.05
Morrow MD.1	CP/M-80	ASM	19/02/85	4.05

Nokia M.Mikko	CP/M-80	ASM	19/02/85	4.05
N/S Horizon	CP/M-80	ASM	19/02/85	4.05
Ohio Sci.	CP/M-80	ASM	19/02/85	4.05
Osborne 1	CP/M-80	ASM	19/02/85	4.05
S/brain	CP/M-80	ASM	19/02/85	4.05
T.Zorba	CP/M-80	ASM	19/02/85	4.05
TRS80-II	CP/M-80	ASM	19/02/85	4.05
Vec.Grap.	CP/M-80	ASM	19/02/85	4.05
Xerox 820	CP/M-80	ASM	19/02/85	4.05
Fujitsu 16	CP/M-86	ASM86	15/02/85	2.9
Honeywell L6/10	MS-DOS	MASM	05/10/84	1.20A
HP 150	MS-DOS	HP 150		
HP 98xx	UCSD p-system	HP Pascal	20/01/84	
DEC PRO350	P/OS,PRO/RT	Macro 11	20/03/85	
TRS80/4	TRSDOS	ASM	21/03/85	4.0
Intel Dev.Sys.	ISIS	PL/M	04/04/85	
Sanyo MBC	MS-DOS/PC-DOS	MASM	10/06/85	2.28
H/Z-100	MS-DOS	MASM	10/06/85	2.28
NEC APC	MS-DOS	MASM	10/06/85	2.28
Generic	MS-DOS	MASM	10/06/85	2.28
IBM PC	MS-DOS	MASM	10/06/85	2.28
Rainbow100	MS-DOS	MASM	10/06/85	2.28
ACT Apricot	MS-DOS	MASM		2.27
HP-150	MS-DOS	MASM	10/06/85	2.28
HP-110	MS-DOS	MASM	10/06/85	2.28
Wang PC	MS-DOS	MASM	10/06/85	2.28
TI-Pro	MS-DOS	MASM	10/06/85	2.28
ICL Perq	PERQ OS	Pascal	04/12/84	2.0
DEC pro 350	p/os	Bliss	01/06/84	1.0
Rainbow binaries	CP/M-86	ASM86	03/12/84	2.9
Sirius 1	MS-DOS	MASM	12/06/84	1.20
Seequa Chameleon	MS-DOS	MASM	17/11/83	1.18
DEC VT180	CP/M-80	Turbo Pascal	13/02/85	1.1
TRS80 I and III	MS-DOS	Z80 assembler	08/08/84	3.5
Tektronix 4170	CP/M-86	ASM86	03/12/84	2.9
IBM-PC	UCSD p-sys IV.x	Pascal	23/05/84	0.1
Pascal m/engine	UCSD p-sys	Pascal	03/12/84	III.0
Terak 8501a	UCSD p-sys II.0	Pascal, Macro 11	17/11/83	
Sirius 1		CI C-86	07/09/84	1.0
Sirius 1	CP/M-86	ASM86	25/11/83	1.1
Sirius 1	MS-DOS	MASM	10/11/83	1.18

8: ACKNOWLEDGEMENT

I would like to thank Alan Phillips of Lancaster University for his help in preparing this document.

9: AUTHORSHIP

Kermit for EMAS was written by Adam D. Albert-Recht, and is maintained by Christopher J. Adie, ERCC. Queries regarding this software should go to the ERCC Advisory service in the first instance.

Appendix I: Comparison of Costs and Timings for X-Talk and KERMIT

The test file was ASCII text, 50000 bytes long. All timings are approximate, and measured in seconds.

EMAS

(Connection through a PAD at 9600 baud)

	KERMIT	X-TALK
Elapsed Time:	447	87
CPU + PT/650:	89.08	13.51

ERCVAX

(Connection through a PAD at 9600 baud)

	KERMIT	X-TALK
Elapsed Time:	316	N/A
CPU Time:	28.58	N/A

(Directly connected at 9600 baud)

	KERMIT	X-TALK
Elapsed Time:	107	186
CPU Time:	26.68	99.23

INDEX