APPLE II KERMIT USER GUIDE

Version 3.87

Ted Medin, NOSC

1990 Oct 10

1. Apple II Kermit

Authors: Antonino N. J. Mione (Stevens Institute of Technology),
Peter Trei (Columbia University),
Ted Medin (NOSC),
Bob Holley (SERDAC)

Version: 3.87

Date: 1990 Oct

Kermit-65 Capabilities At A Glance:

Local operation:	Yes
Remote operation:	Yes
Transfers text files:	Yes
Transfers binary files:	Yes
Wildcard send:	Yes
^X/^Y interruption(Q):	Yes
Filename collision avoidance:	Yes
Can time out:	Yes
8th-bit prefixing:	Yes
Repeat count prefixing:	No
Alternate block checks:	No
Terminal emulation:	Yes (VT52, VT100)
Communication settings:	Yes
Transmit BREAK:	Yes
IBM communication:	Yes
Transaction logging:	No
Session logging (raw download):	Yes
Raw upload:	No
Act as server:	Yes
Talk to server:	Yes
Advanced commands for servers:	Yes
Long packets:	Yes
Sliding windows:	No
Local file management:	Yes
Handle file attributes:	Yes
Command/init files:	Yes
Printer control:	Yes

Kermit-65 is a program that implements the Kermit file transfer protocol for the Motorola 6502 processor family (hence the name, Kermit-65) on the Apple II microcomputer system. It is written in 6502 assembly language and should run on any Apple II or compatible running DOS 3.3 or PRODOS. This section will describe the things you should know about the file system in order to make effective use of Kermit, and then it will describe the special features of the Kermit-65 program.

1.1. Supported Systems and Devices

There are several different Apple II's which can run Kermit-65. Kermit will have no problems running on an Apple II, II+, //e, //c or //gs system. Of the different communication devices available for the Apple II, Kermit-65 supports the ones shown in Table 1-1.

It is possible that other cards may have operational characteristics very similar or identical to one of the devices above. If this is the case, it may work using one of the currently available device drivers. The user may want to try each of the above options to see if any of them work. Kermit-65 must be told in which slot the card resides. This may be done with the 'SET' command (documented below).

AE Serial Pro (super serial driver - sw 1 & 3 open 2 & 4 closed)
AIO II (Uses the Apple Com Card driver??? - untested)
ALS dispatcher (Uses the Apple Com Card driver)
Apple Cat Serial Card
Apple Com Serial Card
ASIO (Uses the Apple Com Card driver??? - untested)
Apple Super Serial Card & //c Serial Port
Apple //gs Serial Port
CCS 7710 Serial Card
CCS 7711 (Uses the Apple Com Card driver??? - untested)
D.C. Hayes Micromodem.
Microtek sv-622 Card
Prometheus Versacard (Uses the Apple Com Card driver)
SSM AIO (Uses the Apple Com Card driver??? - untested)

Table 1-1: Apple II Communication Cards Supported by Kermit-65

1.2. The DOS 3.3 File System

Items of importance which will be discussed in this section include filenames and file characteristics.

Apple DOS Filenames

Filenames under Apple DOS may contain almost any ASCII character (including space). It is not recommended that special characters, (i.e. control characters or spaces) be used in a filename to be transferred by Kermit-65 since they may cause problems when parsing the filename. Filenames may be up to 40 characters in length.

Apple DOS File Characteristics

All files in Apple DOS have a file type associated with them which is contained in the directory entry for the file but is not part of the filename itself. There are four types of files in DOS 3.3. They are:

1. APPLESOFT BASIC
2. INTEGER BASIC
3. BINARY
4. TEXT

All file types have their data stored in eight-bit bytes although not all of them need the eighth bit. The two file types containing basic programs required the eighth bit due to the nature of the data being stored. BINARY files are images of memory copied into a file. Often, these are machine code programs. These files require all eight bits. TEXT files normally contain only printable or carriage control characters. They are stored in the form of seven-bit ASCII characters but the eighth bit should always be set since Apples manipulate all text internally as 'Negative ASCII'. When transmitting non-text files the user must insure that both Kermits are handling eight-bit data so that no information is lost. If an eight-bit data path is not available (i.e. the remote Kermit needs to do parity checking with the eighth bit), then eight-bit quoting should be used. Of course, BINARY files as well as Apple BASIC files will not have much meaning on a different system. If the user desires to edit a BASIC file on a mainframe, for instance, s/he must convert it to a TEXT file before sending it over. After receiving the file back on the Apple, the user may convert it back to BASIC once again. The reason BASIC files would be meaningless to a different machine is that the Apple stores BASIC keywords as single character tokens to save space and processing time. To convert a BASIC program to and from a TEXT file, consult the Apple DOS 3.3 Manual. File information can be obtained by issuing the CATALOG command. For example:

```
]CATALOG
```

```
 DISK VOLUME 010

   *A 002 HELLO
    B 078 KERMIT
    A 002 READER
    T 005 TESTFILE

]
```

When Kermit-65 is receiving a file, the file it creates on diskette will be of the type indicated by the FILE-TYPE parameter. The file will always be left in an unlocked state after it is closed by Kermit-65. When sending a file, Kermit-65 will use the FILE-TYPE parameter to determine how to detect an End-of-file condition. Thus, it is important to have this set properly in all cases.

Recommendations for Archiving Files

When using a large system for archiving purposes, there is no reason to convert Apple Basic programs into text files before sending them if there is no need to edit them on the mainframe. The FILE-TYPE parameter must always be set correctly when sending and receiving files. The procedure for archiving files is:

1. Run Kermit on remote system.
2. `SET FILE-TYPE TEXT` (*or* `APPLESOFT` *or* `...`) on Kermit-65.
3. Send the files.

1.3. The PRODOS File System

The PRODOS system is essentially the same as the DOS system with the exception that performance has been improved, hardware usage has been expanded and file names have different syntax. File names are the major importance to the Kermit system. File names have the following syntax:

```
/volname/subdirectory1/.../subdirectoryn/filename
```

where "volname" is the volume name where the file is located. Subdirectory(n) is a subdirectory on the volume and may be omitted. Filenames are much more restrictive than DOS filenames. PRODOS filenames are limited to 15 characters with no embedded spaces and few special characters, and must begin with an alphabetic character. /volname/sub ... may be omitted from the filename by use of the SET PREFIX command.

Binary file transfer using PRODOS has its dangers when creating new files. PRODOS keeps the file’s size and starting location in the directory which is of course not transferred. Therefore a new binary file will have its starting location 0 which can cause some interesting problems if you try and BRUN the file. Basic files all start at $801 (it says here) so Kermit creates new basic files with a starting address of $801.

1.4. Program Operation

Prior to using Kermit-65 for transferring files, the modem interface must be set to handle data in a certain manner. First, the data format should be 8 data bits and 1 stop bit. Second, the card should be set to no parity. The baud rate (if adjustable) must be set to whatever rate the modem can handle. For the D.C. Hayes Micromodem, these parameters are set correctly by default, so very little has to be done. For the Apple Super Serial Card these are set from within Kermit-65 except the interrupt switch (sw6-2) which must be set for interrupts on. For the Microtek SV-622, all applicable parameters are set by Kermit-65. Some mainframes may need parity checking (i.e. most IBM machines). In this case some parity setting (other than none) will usually work. When talking with such mainframes, binary and basic files on the Apple cannot be transferred unless Eighth-bit-quoting is acceptable to the host. If you have the parameters set correctly then the "CONNECT" command will start Kermit talking out the communication port.

File transfer is very dependent upon parity. Make sure the host and local parity are the same. Following are a couple of site's method for file transfer.

We have an IBM 3033 and 4381 and use both 3705/3725 and 7171 or Series/1 front ends. The differences in front ends as far as any microcomputer Kermit is concerned duplex (local-echo on for the 3705, local-echo off for the 7171 or Series/1), parity (the two front ends might use different parity, e.g. Mark for the 3705 and Even for the 7171), and flow control (None for the 3705, XON/XOFF for the 7171).

In Kermit-65, IBM mainframe users need to set the following parameters:

BAUD	Whatever is supported.
PARITY	EVEN, ODD, or MARK, whatever your front end requires.
FLOW	XON for the 7171, NONE for the 3705.
FLOW DELAY	00
LOCAL-ECHO	OFF for 7171, ON for 3705

In Kermit-65, SERDAC VAX 8800 users need to set the following parameters:

BAUD	SERDAC Dial-up & 300, 1200, or 2400 baud FIRN Dialup:(the highest your modem and the dial-up connection will support)) Ethernet Hardwire: 300, 1200, 2400, or 4800 baud.
PARITY	NONE
FLOW	XON
FLOW DELAY	00 (higher for printers, logging, or "slow" Apples)
LOCAL-ECHO	OFF

NOTE: If you want to do a binary file transfer (Apple binary or BASIC files) via a FIRN Network connection to the SERDAC VAX 8800, you must SET PARITY SPACE before the transfer is initiated; that will insure that eight-bit quoting is used. If you dial directly into the VAX 8800, SET PARITY NONE; eight-bit quoting (which is less efficient) is not required.

Conversing With Kermit-65

Kermit-65 reads file `KERMIT.INIT` from the default drive when started. The lines of this file are executed one at a time starting at the beginning. This file should be an ASCII text file and contain commands to set up Kermit's parameters as desired. It will also execute Kermit's other commands. However, any command which reads a file (like MODEM) or leaves local mode (like CONNECT) will terminate reading of this file and continue with the command specified. Use your favorite editor to produce this file. Here's a sample:

```
set display 80 3
set keyboard 2e
set baud 4800
modem
```

Kermit-65's prompt is "`Kermit-65>`". To run Kermit-65 and issue commands to it, type "`brun kermit`". Example:

```
]BRUN KERMIT

NOSC/STEVENS/CU - APPLE ][ KERMIT-65 - VER 3.87

Kermit-65>send testfile

    (file is sent...)

Kermit-65>status

    (performance statistics are printed...)
```

```
Kermit-65>(other commands...)
          .
          .
          .
Kermit-65>exit
]
```

Like many Kermit programs, Kermit-65 uses a DEC-20 style command parser. During interactive operation, you may use the ?-prompting help feature ("?") and recognition (ESC) features while typing commands. A question mark typed at any point in a command displays the options available at that point; typing an ESC character causes the current keyword to be completed (or default value to be supplied). If you have not typed sufficient characters to uniquely specify the keyword (or if there is no default value) then a beep will be sounded and you may continue typing. Keywords may be abbreviated to any prefix that is unique.

Remote and Local Operation

Kermit-65 is normally run in local mode. It may be run as a remote Kermit as well although there is no advantage to doing things that way. Kermit-65 supports User-mode commands for talking to a Server, and it does support a limited server mode.

1.5. Kermit-65 Commands

1.5.1. The CATALOG Command

Syntax: `CATALOG`
or `LS`

Typing CATALOG produces a catalog (directory) listing of your default drive.

1.5.2. The CONNECT Command

Syntax: `CONNECT`

Establish a terminal connection to the remote system using all the current SET parameters for terminal type, speed, parity, etc. Get back to Kermit-65 by typing the escape character followed by the letter C. The escape character is Control-`@` by default. When you type the escape character, several single-character commands are possible. These are shown in Table 1-2.

You can use the SET ESCAPE command to define a different escape character. When CONNECTed, Kermit-65 will be passing characters entered on the keyboard to the remote system, and passing characters from the remote system to the Apple screen. Incoming characters are interpreted according the selected terminal type (see SET TERMINAL).

On an Apple II+ with an incomplete keyboard, special characters can be typed by prefixing regular characters with a right-arrow. On uppercase-only screens, uppercase characters are shown in inverse and lowercase characters are displayed as normal uppercase characters.

Here are the rules for using the special 2/2+ input, to get all printable ASCII characters, and how they appear on the screen. Special meanings are applied in various contexts to certain characters. The left and right arrow keys do special things, and sometimes the escape key does as well. For letters, the keyboard is always in either default UPPERCASE mode or default lowercase mode. When in UPPERCASE, all letters typed are sent out as uppercase. In lowercase, all letters are sent as lowercase. To reverse the case for the next character only, hit the right-arrow ("prefix") key. To switch the default case, hit the prefix-key twice in a row. For funny characters, the prefix key is

`?`	List all the possible single-character arguments.
`B`	send a Break signal.
`C`	Command mode (returning to kermit-65).
`D`	Drop the phone line to the remote and return to Kermit-65.
`E`	Erase the screen (useful for clearing garbage on screen).
`K`	toggle Keypad application-mode on/off.
`M`	execute the Modem command.
`P`	toggle the Printer on/off.
`Q`	execute the Quit command.
`R`	pRint the screen, >= //e required
`S`	show Status of the connection.
`V`	cursor-keys-Vt100 toggle.
`W`	sWap the del and backspace key.
`0`	send a null (ASCII 0).
`^@`	(or whatever the Connect-Escape character is): send the Connect-Escape character itself.

Table 1-2: Kermit-65 Single-Character CONNECT Escape Commands

also used to get the unusual punctuation characters which are not on the Apple keyboard. Table 1-3 shows the Apple II/II+ keyboard escapes; the letter "p" represents the prefix character.

To Get	Type	Appearance
Left Square Bracket	`p(`	`[`
Right Square Bracket	`p)`	`]`
Left Curly Bracket	`p<`	`{`
Right Curly Bracket	`p>`	`}`
Underline	`p-`	`_`
Backslash	`p/`	`\`
Tilde (wiggle)	`p^`	`~`
Vertical Line	`p.`	`|`

Table 1-3: Apple II/II+ Keyboard Escapes

The left-arrow key sends a rubout (ASCII 127). With left-arrow and right arrow doing special things, it's a little hard to enter their characters (`^H` and `^U` respectively). There is therefore an escape from prefix mode sequence. If you type prefix-ESC, the next character is sent without any interpretation. If you have the capability for upper/lower case, etc, then use the 'SET KEYBOARD' and 'SET DISPLAY' commands to specify complete keyboards.

While in connect mode if you have a //e or better with 80 column display, the cursor will blink. The rate of blink is tied to the "SET TIMING" constant. Also the screen will be restored to state of the previous connect(if any).

1.5.3. The DELETE Command

Syntax: `DELETE` *filespec*
or `RM` *filespec*

Typing DELETE causes the file specified to be deleted.

1.5.4. The EXIT and QUIT Commands

Syntax: `EXIT` *or* `QUIT`

Exit from Kermit-65. When using dos 3.3 you can probably restart the program, provided you haven’t run anything else, by typing ’CALL 4096’.

1.5.5. The GET Command

Syntax: `GET` *remote-filespec*[,*local-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 with a Kermit server on the other end. The remote filespec is any string that can be a legal file specification for the remote system; it is not parsed or validated locally. So if the remote Kermit supports wildcards you can specify them in the *remote-filespec*. Local-filespec is optional and is the file name to be used locally. The "\" escape character may be used to accept the next character of the filespec asis and two escape characters will parse to a single "\". 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. Currently, a packet can be retransmitted manually by typing anything on the keyboard. If a ’Q’ is typed, the entire transmission will be canceled. During file transfer if the remote kermit supports file attributes then the percent of the file transferred will be accurate else 0.

1.5.6. The HELP Command

Syntax: `HELP`

Typing HELP alone prints a brief summary of the Kermit-65 commands.

1.5.7. The LOCK Command

Syntax: `LOCK` *filespec*

LOCK will file lock the given filespec on the default drive.

1.5.8. The LOG Command

Syntax: `LOG` *filespec*

When connected to a remote site, log the remote session’s output to the specified file. The file type and file warning protocols are observed. This command is dependent upon the flow control (XON/XOFF) working. Without flow control there is little possibility of getting a correct copy of the terminal session. The logging begins when you connect to the remote and is terminated when you escape back to the local Kermit with the ESCAPE character followed by the "C" command.

1.5.9. The MODEM Command

Syntax: *MODEM*

This command is designed for the hayes smart modem. Typing MODEM causes the file `KERMIT.MODEM` in the default drive/path to be used as a menu. You will be able to select any line in the file to be sent to the modem. Sorry, you can't back up to a previous menu, you will have to Quit and execute MODEM again. A "CONNECT" response from the smart modem will cause Kermit to leave the modem command and execute the CONNECT command. The Hayes smart modem must reply with text status responses (not numbers). One command per line with comments allowed after the first space (blank). Use your favorite editor to produce this ASCII text file. Since the attention Hayes command (AT) requires a delay the ''&'' character becomes the time delay for Kermit. Each ''&'' causes a delay of one second on a 6502 chip. If you have a //gs or an accelerator board you may have to use the SET TIMING command to produce a one second delay. If you really need to send the "&" character to the modem then the "\" is the escape character. Put a "\" before any character and that character will be sent as is. Of course two "\"'s will produce one "\". Normally Kermit will wait for 27 seconds (again on a 6502 chip) for the modem to respond, but any character typed on the keyboard will terminate this wait. You may hear the busy signal and there is no sense waiting any longer, so hit (ouch!-not so hard) any key on the keyboard.

Following is an example of the `KERMIT.MODEM` file:

```
+++&&ATH    Get the Hayes Smartmodem's attention and then hang up.
ATDP1234567 Call your local BBS with pulse dialing.
ATDT8901234 Call your work dialup phone with touch tone dialing.
```

1.5.10. The RECEIVE Command

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

The RECEIVE command tells Kermit-65 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 filename, Kermit-65 will attempt to change it into something legal. If FILE-WARNING is on and an incoming file has a name identical to a file already existing on the diskette, Kermit-65 will issue a warning to the user and attempt to modify the filename to make it unique. Currently, a packet can be retransmitted manually by typing anything on the keyboard. If a 'Q' is typed, the entire transmission will be aborted. During file transfer if the remote kermit supports file attributes then the percent of the file transferred will be accurate else 0. Filespec is required when xmodem protocol is used.

1.5.11. The REMOTE Command

Syntax: `REMOTE [`*option character-string*`]`

The only option currently is "kermit". This command submits the command "character-string" to the remote Kermit's command processor. Long replies are not paged so you will have to use ^S to stop the screen. The obvious usage is for setting and showing parameters on the remote Kermit.

1.5.12. The RENAME Command

Syntax: `RENAME` *filespec,new-filespec*

RENAME will rename filespec to new-filespec on the default drive.

1.5.13. The SEND Command

Syntax: `SEND` *filespec*[,*remote-filespec*]

The SEND command causes a file to be sent from the Apple to the remote system. The Filespec is the name of the file on the Apple diskette to be sent. The parser will not accept control characters and certain special characters in a filename (like comma). The "\" escape character may be used to accept the next character of the filespec asis and two escape characters will parse to a single "\". Remote-filespec is optional and is the name of the file on the remote kermit. Thanks to Dick Atlee, wildcards are now acceptable when sending files (they have always been acceptable when receiving files). The ‘‘*’’ is a multiple character wildcard and the ‘‘=’’ is a single character wildcard.

The default disk drive is used for file transfers this can be changed with the ’SET DEFAULT-DISK’(DOS) or ’SET PREFIX’(PRODOS) command (explained below). As a file is being sent, the screen displays ’RECEIVING NUMBER OF BYTES’ and ’SENDING NUMBER OF BYTES’ with the decimal number of bytes transferred since start of transmission. If a packet must be transmitted several times and it reaches the maximum retry count, the transfer will fail and the ’Kermit-65>’ prompt will return. If the remote Kermit sends an error packet, the text of the packet will be displayed on the screen, the transfer will fail, and the prompt will return. Currently, a packet can be retransmitted manually by typing anything on the keyboard. If a ’Q’ is typed, the entire transmission will be aborted.

1.5.14. The SERVER Command

Syntax: `SERVER`

Typing SERVER alone turns Kermit into a file server to a remote Kermit. Currently server mode will handle remote "send", "get", "remote" and "fin" commands. Variants of the above commands will probably work but file serving is very limited at present. Because the Apple requires knowledge of file types you can use the "remote Kermit" (or whatever the remote Kermit’s syntax is) command to set the file-type on the server. Yes, the server will execute any command so you can really get the server into trouble (this is not a BBS). You must have the appropriate file type set before transferring files. You can exit server mode by typing Control-C (`^C`) when not doing file transfers or the remote can of course terminate via the "fin" command.

1.5.15. 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:

APPLICATION-MODE	Set VT100 gs keypad in/out of application mode.
BAUD	Which baud rate should the com card use?
CLEAR-SCREEN	Should screen be cleared when returning from connect?
CURSOR-KEYS-VT100	In VT100 mode cursor keys give VT100 sequences.
DEBUGGING	TERSE or VERBOSE packet information.
DEFAULT-DISK	Which Diskette drive is used for DOS 3.3 file transfer?
DISPLAY	Which type of screen display is being used?
ESCAPE	Character for terminal connection.
FILE-TYPE	Type of Apple file being sent/received.
FILE-WARNING	Warn users if incoming file exists?
FLOW	Should xon/xoff flow control be used to the remote?
KEYBOARD	II+ or //e keyboard.
LOCAL-ECHO	Full or half duplex switch.
PARITY	Character parity to use
PREFIX	Which default prefix to use with PRODOS?
PRINTER	Should the printer be used for the display?
PROTOCOL	Which protocol is to be used for file transfer.
RECEIVE	Various parameters for receiving files
SEND	Various parameters for sending files
SLOT	Which slot # is communication device in?
SWAP	Swap the del and backspace key?
TIMER	Should Kermit observe the receive timeout value?
TIMING	Change the time loop for 1 ms. delays.
TERMINAL	Which type of terminal should Kermit emulate?

SET APPLICATION-MODE

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

For VT100 emulation with a gs keypad you can set the keypad in or out of application mode. Some computer systems set this via escape sequences so it may not be necessary to use this command.

SET BAUD

Syntax: `SET BAUD` *value*

Value is the baud rate for your communication card. For the super serial and the microtek it can be 300 to 19200. The actual values will depend upon the com card you are running with.

SET CLEAR-SCREEN

Syntax: `SET CLEAR-SCREEN {ON, OFF}`

When returning from a connect the screen will be cleared if on. This is for a //e or better machine.

SET CURSOR-KEYS-VT100

Syntax: `SET CURSOR-KEYS-VT100 {ON, OFF}`

In VT100 emulation the cursor keys can also emulate the VT100 cursor keys.

SET DEBUGGING

Syntax: `SET DEBUGGING {TERSE, VERBOSE, OFF}`

Record the packet traffic on your terminal. Options are: TERSE, Show packet info only (brief). VERBOSE displays packet field descriptions with packet info (lengthy). OFF disables display of debugging information (this is the default).

SET DEFAULT-DISK

Syntax: `SET DEFAULT-DISK {SLOT, VOLUME, DRIVE}` *value*

This DOS command will tell Kermit-65 which disk drive should be used for file transfers. The three parameters which may be set separately are SLOT, VOLUME and DRIVE. The value for SLOT ranges from 1 to 7. The value for DRIVE is either 1 or 2. The value for VOLUME ranges from 0 to 255.

SET DISPLAY

Syntax: `SET DISPLAY {2E, 2P}`
or `SET DISPLAY 80-COL` *number*

This command will tell Kermit-65 which kind of screen display you want to use. If you have an Apple II or II+ without an 80 column card, use the first syntax. If you have any kind of an Apple with an 80 column card, enter: SET DISPLAY 80, followed by a space and the slot number where the card resides (if you don't know the slot number, or the card is built-in to the set, try 3).

SET ESCAPE

Syntax: `SET ESCAPE` *hexadecimal-number*

Specify the control character you want to use to "escape" from remote connections back to Kermit-65. The default is 0 (Control-@). The number is the hex value of the ASCII control character, 1 to 37, for instance 2 is Control-B, B is Control-K.

SET FILE-TYPE

Syntax: `SET FILE-TYPE {APPLESOFT, INTEGER, TEXT, BINARY, OTHER` *hex-value*`}`

This will inform Kermit-65 what type of file is being sent or received. It is important that this is set correctly since Kermit-65 must create a file of the appropriate type when receiving. With the advent of file attributes (if the other kermit does them) this has been improved somewhat but since file attributes usually only knows about text and binary one may endup with the wrong type. So keep your type changing utility handy. When Kermit-65 is sending, it will know the type of file but again be careful of file attributes (assuming the other kermit does them) for the file may end up as binary on the other end. The keywords for this parameter are listed below. OTHER includes an added hex-value so that the user may specify the hex value of the file-type. This has meaning only in PRODOS and allows the user to specify any of the many different file types used in PRODOS, see Tables 1-5 and 1-6 (thanks to Phil Chien, M L Stier et al).

APPLESOFT	The file being transferred is an Applesoft Basic program.
INTEGER	The file being sent/received is an Integer Basic program.
TEXT	The file being sent/received is an ASCII Text file.
BINARY	The file being sent/received is a Binary image.
OTHER	The type of file being sent/received is specified by the hex-value.

SET FILE-WARNING

Syntax: `SET FILE-WARNING {ON, OFF}`

This tells Kermit-65 whether to warn the user about incoming filenames conflicting with existing files or not. If there is a conflict Kermit-65 will attempt to change the file name to something unique.

SET FLOW

Syntax: `SET FLOW {OFF, XON, DELAY` *number*`}`

SET FLOW allows one to use the XON/XOFF protocol when connected to a remote site. Delay timings are part of this command. Using delay times is probably a desperation move to keep the screen/printer from losing characters. Setting the timings will have to be set by experience. Perhaps the best way to set the timings is to bring the values down until you get failures and then double the timing figure. Both LOG and SET PRINTER will probably depend on flow control.

OFF	Turn off flow control
XON	Turn on xon/xoff flow control with the remote
DELAY *number*	Delay the micro until XOFF takes effect

Delay followed by a number (including 0) delays the program for that many milliseconds after the XOFF is given to the remote. This delay allows the XOFF to take effect before the program continues.

NOTE: Except for printing and logging, most Apples will not require you to use a flow delay, even at rates up thru 19200 baud; for proper screen control, however, certain older Apple IIe's may require a fairly high delay (120-160 dec), even at 300 baud.

SET KEYBOARD

Syntax: `SET KEYBOARD {2P, 2E}`

SET KEYBOARD tells Kermit-65 if the user has a full keyboard (2E) or not (2P). If the user is on an Apple II+, this should be set to 2P (which is the default). When set to that, character translations are available by using the right-arrow key as a prefix character, as shown in Table 1-3.

SET LOCAL-ECHO

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

This command tells Kermit-65 to echo to the screen characters you type on the keyboard (LOCAL-ECHO = ON), or to let the remote system echo the typed characters (LOCAL-ECHO = OFF). If, when CONNECTed to the remote, you see a duplicate of every character you type, escape back to Kermit-65, and SET LOCAL-ECHO OFF. If, when CONNECTed to the remote, you see nothing echoed to the screen, escape back to Kermit-65, and SET LOCAL-ECHO ON.

SET PARITY

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

This command tells Kermit-65 which parity you want to use while communicating with the remote. Most remotes use NONE; some use EVEN, a few may use the other possible values. If you have a choice of parity to use with a remote machine, if possible, choose NONE.

SET PREFIX

Syntax: `SET PREFIX string [Default: boot volume]` *or* `CD string`

This command allows you to specify a ProDOS volume/file prefix.

SET PRINTER

Syntax: `SET PRINTER {ON, SLOT}` *number*
or `SET PRINTER OFF`

This allows one to turn the printer on for printing what is displayed on the screen. The printer can also be toggled on/off via the ESCAPE character followed by the command "P".

Remember when you use your printer there are a lot of variables here. What was being sent to the screen now is being sent to your printer. If you were emulating the VT52 your printer may not know how to handle the escape sequences, tabs, etc. It may be you can tell the host you are a tty or some such device that will produce control codes that your printer can handle. Some printers may require the flow control and delay to get readable printing.

ON	Turn the printer on, slot number is required.
OFF	Turn the printer off.
SLOT *number*	Printer card is in slot "number".

SET PROTOCOL

Syntax: `SET PROTOCOL {KERMIT, XMODEM}`

SET PROTOCOL tells kermit-65 which protocol to use for file transfer. NOTE: When XMODEM is used you will probably want to change the carriage return and carriage return/line feed translation in the send/receive parameters. eg. "SET SEND CR<->CR,LF OFF" and "SET RECEIVE CR<->CR,LF OFF".

SET RECEIVE

Syntax: `SET RECEIVE {CR-CR,LR, EIGHT-BIT-QUOTE, END-OF-LINE, PACKET-LENGTH, PAD-CHAR, PADDING, QUOTE-CHAR, START-OF-PACKET, TIMEOUT}`

This will inform Kermit what to use to form and handle receive packets for file transfer.

CR<->CR,LF	{ON, OFF} Terminate lines with cr or cr and lf.
EIGHT-BIT-QUOTE	*hexadecimal-number* Char for eighth bit quoting.
END-OF-LINE	*hexadecimal-number* Char for line termination.
PACKET-LENGTH	*hexadecimal-number* Size of packet.
PAD-CHAR	*hexadecimal-number* Char for padding.
PADDING	*hexadecimal-number* Number of padding chs.
QUOTE-CHAR	*hexadecimal-number* Char for quoting.
START-OF-PACKET	*hexadecimal-number*

	Char for start of packet.
TIMEOUT	*hexadecimal-number* Number of seconds for timeout.

SET SEND

Syntax: `SET SEND {CR-CR,LR, EIGHT-BIT-QUOTE, END-OF-LINE, PACKET-LENGTH, PAD-CHAR, PADDING, QUOTE-CHAR, START-OF-PACKET, TIMEOUT}`

This will inform Kermit what to use to form and handle send packets for file transfer. The options are the same as the "SET RECEIVE ..." packets.

SET SLOT

Syntax: `SET SLOT` *number*

This option tells Kermit-65 in which slot the communication device is located. The range for the number parameter is 1-7.

SET SWAP

Syntax: `SET SWAP {ON, OFF}`

This option tells Kermit-65 to swap the functions of the del and backspace keys.

SET TIMER

Syntax: `SET TIMER {ON, OFF}`

SET TIMER will turn on or off the timeout checking for receive file transfers. Since there is no clock for exact timing a loop of instructions has been set up assuming a 1 megacycle CPU. CPUs which run faster may have to make allowances via the SET RECEIVE TIMEOUT command or the SET TIMING command.

SET TIMING

Syntax: `SET TIMING { number }`

Kermit uses a timing loop with the rom address $fca8 to produce a 1 ms. delay. If you have a machine that runs faster than the 6502 chip you may have to increase this number to get the 1 ms delay.

SET TERMINAL

Syntax: `SET TERMINAL {MONITOR, NONE, VT100, VT52}`

When TERMINAL is NONE, then all incoming characters (except nulls) are passed directly to the display.

MONITOR emulation simply displays all the characters received from the remote (except nulls) without any formatting of the screen (40 or 80 characters per line). Control characters are displayed inverse.

VT100 Emulation

The Kermit-65 VT100 emulator is a small but working set of a true VT100 terminal. It appears to work with most of the standard full screen editors and processors on BSD UNIX and VAX/VMS machines. An Apple//e, //c, or //gs is probably required with the Apple 80 column text card. The VT100 keypad has also been defined for the application mode via the OA/CA/game button. Figure 1-1 shows the vt100 keypad on an apple keyboard with EDIT (VMS) usage, and Figure 1-2 shows the layout on an Apple//gs keypad. When using EVE (VMS) the meaning of the keys will of course change.

As you can see the keypad is physically laid out like the VT100 keypad except for the lower right corner. Notice that above the keys are the VT100 labels while in middle of the box (key) is the Apple key label. Also the arrow keys work as VT100 arrow keys with the OA/CA/game button.

```
                                PF1     PF2     PF3     PF4     .(period)
                              ------- ------- ------- ------- -------
                             |       |       |findnxt|delline|select |
                             |   6   |   7   |   8   |   9   |   0   |
                             | gold  | help  | find  |undelln| reset |
                              ------- ------- ------- ------- -------
                                  7       8       9       -
                                ------- ------- ------- -------
                               | page  |section|append |delword|
                               |   Y   |   U   |   I   |   O   |
                               |command| fill  |replace|undelwd|
                                ------- ------- ------- -------
     vt100 key ---->                 4       5       6       ,
                                   ------- ------- ------- -------
     vms edit usage ---->         |advanc|backup |  cut  |delchar|
     apple key ---->              |  H   |   J   |   K   |   L   |
     vms edit usage ---->         |bottom|  top  | paste |undelch|
                                   ------- ------- ------- -------
                                       1       2       3      ENTER
                                     ------- ------- ------- -------
                                    | word  |  eol  | char  | enter |
                                    |   N   |   M   |   ,   |   .   |
                                    |chgcase|deleol |specins|  sub  |
                                     ------- ------- ------- -------
                                         0
          ---------------------------------------
                                          bline       |
               space bar                              |
                                         openline     |
          ---------------------------------------
```

Figure 1-1: VT100 Keypad on an Apple Keyboard

VT52 Emulation

SET TERMINAL VT52 will turn on the VT52 terminal emulation. One thing that is required is your 80-column card must handle the $16 command in order for reverse scrolling to work. The Apple//e 80 column card handles this fine. The VT52 keypad has been defined using the open/closed Apple. For II or II+ one will have to have a game paddle or joy stick (key shift mod too????) and use the buttons. When a button/open/closed Apple is pushed then the keys starting with 6,7,8 & 9 form the top of the keypad. Key 6 is the blue key key 7 is the red key etc. The keys directly below the 6,7,8 & 9 and shifted one-half key to the right form the second row of the keypad etc. Every thing is fine until you get to the last row on the keypad. There the sp bar is 0 and the other two keys are moved to the upper right as the 0 & - keys. This way the arrow keys are available as VT52 keys with the OA/CA/game button combination (thanks to Dick Atlee for this idea). With those two exceptions the keypad is physically similar to a VT52 keypad. Remember the open/closed Apple or the game button must be pushed (like the control key) to get the keypad emulation. Figure 1-3 should clear up the questions.

```
                    PF1     PF2     PF3     PF4
                  ------- ------- ------- -------
                 |       |       |findnxt|delline|
                 | CLEAR |   =   |   /   |   *   |
                 | gold  | help  | find  |undelln|
                  ------- ------- ------- -------
                   7       8       9       -
                  ------- ------- ------- -------
                 | page  |section|append |delword|
                 |   7   |   8   |   9   |   +   |
                 |command| fill  |replace|undelwd|
                  ------- ------- ------- -------
vt100 key ---->      4       5       6       ,
                  ------- ------- ------- -------
vms edit usage ->| advanc|backup |  cut  |delchar|
gs key ------->  |   4   |   5   |   6   |   -   |
vms edit usage ->| bottom|  top  | paste |undelch|
                  ------- ------- ------- -------
                   1       2       3       ENTER
                  ------- ------- ------- -------
                 | word  |  eol  | char  |       |
                 |   1   |   2   |   3   |       |
                 |chgcase|deleol |specins|       |
                  ------- ------- -------| enter |
                         0           .   | ENTER |
                  -----------------------|  sub  |
                 |     bline     |select |       |
                 |       0       |   .   |       |
                 |   openline    | reset |       |
                  -------------------------------
```

Figure 1-2: VT100 Keypad on an Apple//gs

1.5.16. the SHOW command

Syntax: `SHOW [`*option*`]`

The SHOW command displays various information:

ALL	All parameter settings (this is quite long).
BAUD	Baud rate of the com card.
APPLICATION-MODE	Keypad in application mode?
CLEAR-SCREEN	Clear screen on return from connect?
CURSOR-KEYS-VT100	Are the cursor keys emulating the VT100 cursor keys?
DEBUGGING	Debugging mode.
DEFAULT-DISK	Which Diskette drive is used for file transfer?
DEVICE-DRIVER	Which communication device is being used?
DISPLAY	Which screen display is being used?
ESCAPE	Character for terminal connection.
FILE-TYPE	Of Apple DOS/PRODOS file being sent/received.
FILE-WARNING	Warn users if incoming file exists?

```
^ is up arrow    BLUE     RED     GRAY     ^    .(period) ENTER
                 ------- ------- ------- ------- ------- -------
                |       |       |delline|  up   |select | enter |
                |   6   |   7   |   8   |   9   |   0   |   -   |
                | gold  | help  |undelln|replace| reset |  sub  |
                 ------- ------- ------- ------- ------- -------
  v is down arrow     7       8       9       v
                    ------- ------- ------- -------
                   | page  |findnxt|delword| down  |
                   |   Y   |   U   |   I   |   O   |
                   |command| find  |undelwd|section|
                    ------- ------- ------- -------
   vt52 key ---->         4       5       6       ->
                       ------- ------- ------- -------
vms edit usage ---->   |advanc|backup |delchar| right |
   apple key ---->     |  H   |   J   |   K   |   L   |
vms edit usage ---->   |bottom|  top  |undelch|specins|
                       ------- ------- ------- -------
                            1       2       3      <-
                          ------- ------- ------- -------
                         | word  |  eol  |  cut  | left  |
                         |   N   |   M   |   ,   |   .   |
                         |chgcase|deleol | paste |append |
                          ------- ------- ------- -------
                               0
    ---------------------------------------
                                bline       |
         space bar                          |
                               openline     |
    ---------------------------------------
```

Figure 1-3: VT52 Keypad on an Apple Keyboard

FLOW	Should XON/XOFF flow control be used to the remote?
KEYBOARD	II+ or //e keyboard.
LOCAL-ECHO	Full or half duplex switch.
LOG	Are we logging connect data?
PARITY	Character parity to use
PREFIX	Which default prefix to use with PRODOS? Alias PWD.
PRINTER	Should the printer be used for the display?
PROTOCOL	Which protocol is to be used for file transfer.
RECEIVE	Various parameters for receiving files
SEND	Various parameters for sending files
SLOT	Which slot # is communication device in?
SWAP	Swap the del and backspace keys?
TIMER	Is the receive timeout on or off?
TIMING	Count for timing loop
TERMINAL	Which terminal (if any) should Kermit emulate?
VOLUMES	Show the online volumes.

The above options are analogous to the equivalent SET commands.

1.5.17. The STATUS Command

Syntax: `STATUS`

Give statistics about the most recent file transfer. This includes information such as number of characters sent/received, number of data characters sent/received, and last error encountered.

1.5.18. The TAKE Command

Syntax: `TAKE` *filespec*

The TAKE commands tells kermit-65 to execute commands from the specified file similarly to the KERMIT.INIT file. See discussion on KERMIT.INIT above for details.

1.5.19. The TYPE Command

Syntax: `TYPE` *filespec*

The TYPE commands tells kermit-65 to print to the screen from the specified file. Text files only and works best with 80 characters per line or less.

1.5.20. The UNLOCK Command

Syntax: `UNLOCK` *filespec*

UNLOCK will unlock the given filespec on the default drive.

1.6. Installation

1.6.1. Standard Installation

To bootstrap Kermit to prodos, get the file `APPPRO.BNS` on a apple diskette. Use binscii to convert this file into a shrinkit file. Use shrinkit to extract the files onto a prodos diskette with your own prodos and basic.system files. Then "-read.me" for further instructions. Thanks to Les Ferch for this install system & thanks to Bird for his "birds better bye" in the menu.system. For dos 3.3 get the files `APP387.[1-2]` on a master diskette then read the first of app387.1 for further instructions. Thanks to Alan Kalker for this install system.

The kermit binaries will run on either system. Simply use the PRODOS conversion routines (if they have them fixed, better to use copy II+ or whatever) to move `kermit`, `kermit.help` and `kermit.init` files. If you want other options as a regular thing then you can change file `kermit.init` with your favorite editor.

Files Supplied for Kermit-65

The following files should be supplied on the columbia distribution tape:

`APP387.1`	Install system for dos 3.3 (1 of 2)
`APP387.2`	Install system for dos 3.3 (2 of 2)
`APPAAA.HLP`	List of files (like this one)
`APPAAA.NEW`	Whats new in this release
`APPACC.HEX`	Apple com card hex
`APPACC.M65`	Apple com card source

`APPACE.HEX`	Ace dual card hex
`APPACE.M65`	Ace dual card source
`APPBEL.BNS`	Apple gs alternate bell
`APPCAT.HEX`	Apple cat hex
`APPCAT.M65`	Apple cat source
`APPCCS.HEX`	CCS 7710 com card hex
`APPCCS.M65`	CCS 7710 com card source
`APPCPS.HEX`	CPS com card hex
`APPCPS.M65`	CPS com card source
`APPGS.HEX`	GS serial port hex
`APPGS.M65`	GS serial port source
`APPHMM.HEX`	Hayes micro modem card hex
`APPHMM.M65`	Hayes micro modem card source
`APPICON.BNS`	Kermit icon for gs users
`APPLE.DOC`	Complete documentation (it says here)
`APPLE.MSS`	Scribe text formatter source for documentation
`APPLE.PS`	Documentation ready for a postscript printer
`APPMAI.HEX`	Main kermit pgm hex
`APPMAI.M65`	Main kermit pgm source
`APPMAK.UNX`	Make file for UNIX cross assembly (to assemble Kermit)
`APPMSV.HEX`	Microtec com card hex
`APPMSV.M65`	Microtec com card source
`APPPRO.BNS`	Install system for prodos
`APPSSC.HEX`	Super serial com card hex
`APPSSC.M65`	Super serial com card source
`APPXAS.1`	65c02 cross assembler for UNIX system part 1
`APPXAS.2`	65c02 cross assembler for UNIX system part 2
`APPXAS.3`	65c02 cross assembler for UNIX system part 3

The syntax of the filenames may vary. On UNIX systems, the filenames will be in lowercase. On VM/CMS systems, the period will be replaced by a space. All files are text, however the suffix of BNS are binsciied binary files.

1.6.2. Alternate Installation

The main problem exists in getting the hex files onto your diskette as a text file. But again that is a test of your creativity. If you have a version of Kermit running then GET or RECEIVE the file as a text file and you are in business. Since Kermit has been separated into two assemblies then two hex files will have to be present on the diskette. Get the main hex file `APPMAI.HEX` and select which com card hex you will need. First "`exec APPMAI.HEX`". Your Apple (or compatible) will go into monitor and show you `*`'s for several minutes. This is the monitor loading the hex into binary. If you get beeps from the monitor its probably because you didn't get a good copy of the text file. Now EXEC the com card driver you are going to use. You will have to get back into basic(aha another test for you, try "3d0G") to do this. And you will see the monitor loading the com driver. The order of EXEC's is important. The com card should be loaded last. Next get back into basic and do a "`bsave kermit,A$1000,L$7400`". You may have to specify the drive to do this binary save, with a slot or drive on the end of the BSAVE (aha another test). You now run Kermit via "`brun kermit`".

If you want to customize Kermit for your needs, the recommended method is to use file "kermit.init" OR do all your SETs, etc, and then do an "exit". Now you should be back in BASIC. At this point do a "`bsave` *name*`,A$1000,L$7400`" and when you do a "`brun` *name*" all your setups will be remembered. NOTE: If you save your current settings via "`bsave kermit ...`" you may find that moving that binary to another type of Apple (e.g. from a //e to an //e+) will not be possible. So make sure you keep the original binary to move between machine types.

Since the org is now $1000 if you have been using Kermit and then went back to basic for some trivial thing a "CALL 4096" should start up Kermit without having to reload it.

In summary:

1. `EXEC APPMAI.HEX`
2. Choose the com card driver you will use. For example `APPSSC.HEX`.
3. `3D0G`
4. `EXEC APPSSC.HEX`
5. `BSAVE kermit,A$1000,L$7400`

And you should be in business. Remember there is the command HELP and whenever you are into a command a "?" will give you the possible options available at that point of a command. The escape key will finish typing an option if it is possible. The syntax of all the commands and options only requires enough characters to make that command or option unique.

1.7. Problems

Installation

NOTE: When using the super serial driver you must have the cards sw6-2 on. This allows the card to use interrupts. The rest of the switches are set from within Kermit. It appears that you can run your Apple 2 with sw6-2 on and in 99% of the cases will cause no problems. This is because the OS runs with interrupts locked out ("sei" in assembly language) and the program must explicitly give a "cli" for interrupts to work (the super serial driver does).

The AE Serial Pro must have switches 1 & 3 open and 2 & 4 closed. This appears to disagree with the documentation since those settings turn off irq interrupts and turn on nmi interrupts. So watch this it may get corrected in later versions.

The Microtek driver is a super serial look alike which does not run with interrupts. If you have trouble with the super serial driver you might try the MSV driver. For you people with the MSV-622c card, you might try running a jumper from the UART 6551 pin 26 to the card edge pin 30. This will enable interrupts just like the SSC sw2-6, and then you can use the super serial driver.

The Prometheus card will work with the Apple com driver. However you will have to set the switches on the card for baud etc. Evidently this card can not be programmed by the software. If that is not true then here is an opportunity for you to write a better driver. If you do please pass it on for other Prometheus users.

The apple cat uses the modem’s firmware for dialing. Type ";" for a 2 second delay, or any control character to abort the dialing. ROM is not needed if external phone is used for dialing. Supports 110-600 baud, and 45.5 baud (use kermit-65’s 135 baud setting). Supports 1200 baud with apple-cat 212 card, in slot-saver configuration. Allows use of external serial port by typing "X" as dialing string. Thanks to Dick Wotiz for this driver. Reports have it that the Apple Cat will also work with the Apple com driver.

Some have noted the Apple com card must be initialized via the "IN#x" before starting Kermit. Ike has now updated this driver and the initialization is now done within the Apple com driver. Thanks Ike.

Usage

There is the command HELP and whenever you are into a command a "?" will give you the possible options available at that point of a command. The escape key will finish typing an option if it is possible. The syntax of all the commands and options only requires enough characters to make that command or option unique.

When using flow control you may appear to hang. Type a `^Q` (Control-Q) and that may free you up.

Remember when you use your printer there are a lot of variables here. What was being sent to the screen now is being sent to your printer. If you were emulating the VT52 your printer may not know how to handle the escape sequences, tabs etc. It may be you can tell the host you are a tty or some such device that will give carriage returns etc that your printer can handle. Some printers may require the flow control and delay to get readable printing.

File Transfer Errors

"File Transfer Errors," was added to this document by the Southeast Regional Data Center (SERDAC), ’88 July 17.

In spite of the fact that successful Kermit file transfers are almost always error free, there are a number of circumstances which can corrupt, prevent, or interrupt/abort a transfer. In the case of an actual abort, there may be data loss or corruption, and an incomplete file may not have a correct end-of-file. These circumstances may be roughly divided into two groups: (1) problems due to file or disk errors, and (2) problems due to delays or failures in Kermit packet exchange.

Common problems in category (1) include the following:

(a) improper file specification (b) wrong file type (c) protected file(s) (d) disk problems

(1a) problems can occur when you specify, to either the Apple or host Kermit, a non-existent or improperly located file. Misspelling and/or incorrect (sub)directory specification are popular villains! If you are commanding either Kermit to SEND a file (SEND filespec), the problem will be fairly obvious. On the Apple II, you’ll see an error message like: "FILE NOT FOUND." On the VAX/VMS 8800, for example, you’ll see the message: "%KERMIT32 ....., file not found for ’filespec’". In either case, the transfer will not take place. If you’re using Kermit-65 to GET (GET filespec) files from the VAX/VMS Kermit server, and the requested file does not exist in your VAX default directory, you should see a Kermit-32 generated "REMOTE MESSAGE %KERMIT32 ....., file not found for ’filespec’" appear in the transmission status display, and then the Kermit-65 message "CANNOT RECEIVE FILE-HEAD". Transfer of the questionable file will not take place.

(1b) problems can occur if you forget to specify, to either the Apple or host Kermit, what type of file you wish to transfer. If you are using Kermit-65 to send files to a host, you are fairly well protected against this error. If you attempt to send a file whose CATALOG type does not match the FILE-TYPE parameter setting, you will receive a "INCOMPATIBLE FILE FORMAT" error message, or something similar, and the transfer will not take place. If, however, you are receiving (via RECEIVE or GET) a file whose native type does not match the FILE-TYPE setting, the file WILL be received. It will be mis-typed (according to the FILE-TYPE setting), though, and any later attempt to use it on the Apple will probably be unsuccessful.

The same sort of circumstances generally apply for a host Kermit. With the VAX 8800, for example, when Kermit-32 is sending a file, you generally need not worry about setting its file type. When Kermit-32 is receiving a file, however, properly setting its file type is very critical. If you wish to put Kermit-32 in server mode to receive multiple files, set the file type BEFORE using the SERVER command, and make sure that you only send it the appropriate type of files during that server session. You cannot switch file types DURING a given server session!

NOTE: One other way you can get into trouble with "wrong file type" is by trying to send a file which is mixed--mostly text, but with some embedded characters that are not true 7-bit ASCII (i.e., ASCII codes 00-127). This often happens when you are trying to transfer a file which is word processor output. Most word processing software claims to allow you to output a true ASCII or text file, but in some cases it really does not, and in others the choice of output options is confusing. If you have set up either Kermit program to send/receive a text file, and you try to transfer illegal ASCII characters (codes 128-256), your transfer may "hang" or be aborted. At the very best, if the transfer "works," the suspect characters will later probably be meaningless or confusing to the destination machine.

(1c) problems can occur in two ways on the Apple II. If your default drive disk is write protected, and you attempt to receive a file, you will receive a "WRITE PROTECTED" error message, and no transfer will take place. If you have set Kermit-65’s FILE-WARNING parameter to OFF (normally NOT a good idea), and you attempt to receive a file that already exists in a locked state on your default diskette, you will receive a "FILE LOCKED" error message (if the file is very short, you may have to check with a Kermit-65 STATUS command to see the error message), and no transfer will take place.

Similar problems may occur on the host because of various file protection schemes. On the VAX/VMS 8800, for example, Kermit-32 cannot send out a file that you are unauthorized to read. And, it cannot receive a file unless you are authorized to write to that filename and its (sub)directory. If you use Kermit-32 to attempt to SEND (SEND filespec) a protected file, you should see a "%KERMIT32 ....., insufficient privilege or file protection violation for 'filespec'" error message, and no transfer will take place. If you have Kermit-32 in server mode, and you are trying to GET a protected file from it, or you are trying to SEND it a file whose space is protected, you should see a similar Kermit-32 generated REMOTE MESSAGE appear in the transmission status display, and then, on GET, the Kermit-65 message "CANNOT RECEIVE FILE-HEAD". Transfer of the protected file will not take place.

(1d) problems are most likely to occur because of Apple II diskette or drive problems. The following conditions will generate "DISK I/O" or "I/O ERROR" messages when Kermit-65 transfer commands are entered: bad diskette in default drive, no diskette in default drive, default drive door open, and/or unINITialized disk in default drive.

If any of those errors are detected before the attempted transmission of a given file, the transfer of that file will not begin. If any are detected DURING a file transmission, the file transfer will likely abort; at best transmitted data will be incomplete. Data which does reach the destination end of an aborted transfer should be considered very suspect; the disk problem should be corrected and the transfer should be repeated! (The best chance you have for salvaging text file data in an abort is if the file destination is the host machine and you have told its Kermit to save incomplete files, e.g., on the VAX-8800, you need to SET INCOMPLETE KEEP).

One other Apple II disk problem can be encountered while you are using Kermit-65 to receive files. If you exceed the storage capacity of your diskette during a RECEIVE or a GET, you should see a "DISK FULL" error message. Data that has been received up to the point of the overflow will be automatically DELETEd. Make CERTAIN that you do not try to receive any more files until you have DELETEd some files from the problem diskette, or until you have replaced it with one that has adequate capacity to receive the complete file. NOTE: See Section 1.5.4.

It is less likely that (1d) problems would occur because of host machine disk problems. The most likely circumstance you might encounter on the VAX/VMS 8800, for example, would be in receiving a large file and, in the process, exceeding your VAX disk quota. In such a case, you should see an appropriate Kermit-32 generated REMOTE MESSAGE appear within the Kermit-65 transmission status display. If this happens, delete some files from your VAX (sub)directories, and/or have your VAX disk quota increased BEFORE you try the transfer again. If you have issued a SET INCOMPLETE KEEP command to Kermit-32, there may be some chance of salvaging text file data that arrived before the disk quota overage, but the best thing you can do is to repeat the transfer!

As a general rule, if some disk or file error prevents a transfer from beginning, to get it to "go," you will need to correct the error and repeat all the steps that preceded it.

If you are still commanding the host Kermit, and you see an error message, you will have to get the host Kermit's prompt back and give it an acceptable command. If you have commanded the host Kermit to SEND or RECEIVE, and are back commanding Kermit-65 when you notice the error, you will have to correct the problem, CONNECT back to the host, get the host Kermit prompt (with the VAX/VMS 8800, try typing RETURN or CTRL-Y), and repeat the SEND or RECEIVE command, before returning back to Kermit-65 to command it again.

If you have placed the host Kermit in server mode, and are giving Kermit-65 commands when you notice an Apple disk/file error prevents a file transfer from starting, chances are good that you won't have to CONNECT back to the host. It is also important to note that within a single server session, when you are transferring multiple files, all files transferred PRECEDING an error (or abort) are probably good. To repeat the transfer, correct the error, and give Kermit-65 the appropriate command to transfer the file that messed up. The first time you do it, you may get back a message like "REMOTE MESSAGE %KERMIT-32..... protocol error" This is just the host server trying to get back "on track" after the error. When the Kermit-65> prompt returns, enter the transfer command again, and it will probably be accepted.

If the second attempt should fail, wait for the Kermit-65> prompt, enter: FINISH, wait for the prompt again, and enter: CONNECT. If you do not see the host operating system prompt ($ on the VAX 8800), type a few RETURNs (or on the VAX/VMS a CTRL-Y). Re-invoke the host Kermit and put it back in server mode.

If disk or file errors prevent a transfer from completing, recovery will depend on the error, whether you had the host Kermit in server mode or not, and on your desire for accuracy.

Some disk/file error aborts are "fatal" (e.g., Apple DISK FULL, and uploading to the VAX 8800 w/o having commanded Kermit-32 to SET INCOMPLETE KEEP). The destination file will be destroyed. The transfer of the file will have to be repeated again from the beginning. Again, unless you have set the host Kermit for server mode, you will have to CONNECT back, get the host Kermit prompt, and re-command it. If you were in a server session, though, you can probably repeat the transfer of the interrupted file without going back to the host (see recovery procedures above).

Other disk/file errors that interrupt/abort a transfer may leave salvageable text data at the transfer destination. The best policy, though, is to repeat the transfer of the incomplete file (see recovery procedures above).

Common problems in category (2) include the following:

(a) bad parity

(b) noisy communications line

(c) timeout due to delays, "disaster," etc.

(d) Kermit-program incompatibility

(e) user error

(2a)

Parity settings are very critical to correct transfers. If you do not inform Kermit-65 of the correct parity being used by the remote host machine or the communications path to it, "checksum" error checking calculations will be wrong, and packets will be consistently rejected when they arrive at their destination. In particular, most binary file Kermit transfers won't get very far if parity is not set correctly.

[NOTE: If you want to do a binary file transfer (Apple binary or BASIC files) via a FIRN Network connection to the SERDAC VAX/VMS 8800, you must SET PARITY SPACE before the transfer is initiated; that will insure that eight-bit quoting is used. If you dial directly into the VAX/VMS 8800, SET PARITY NONE; eight-bit quoting (which is less efficient) is not required].

(2b)

Line noise can be the root cause for a variety of file transfer problems. The beauty of a "packetized protocol transfer" scheme like Kermit is that ordinarily, the scheme will overcome an occasional burst of line noise. A packet which arrives out of sequence, or which does not have the same checksum "bit count" as when it was sent, will get retransmitted, and the noise induced data error will correct itself.

Sometimes, however, bad line noise can outwit even the cleverest aspects of Kermit. There are some times where severe noise can corrupt the "checksum" error checking and lead to undetected transmission of a bad character (assuming that the severe line noise exists, chances of this happening for one character are, for Kermit-65 error checking, less than two percent).

If line noise is bad enough and persistent enough, it is also a cause for several problems that will eventually "hang" or totally confuse and abort a transfer:

Each transfer is preceded by the Kermit-to-Kermit exchange of several short "initialization packets. These tell the controlling programs critical things to expect about the upcoming transfer. If line noise prevents the packets from arriving, or scrambles them up, the transfer probably can't get started correctly.

One of the biggest vulnerabilities of the Kermit scheme is that each arriving packet must be acknowledged (ACK) by the receiver, and that the sender must actually receive back the acknowledgement (likewise, if an expected packet does not arrive, there often must be a negative acknowledgement (NAK)). Since the ACK/NAK packets are very short, they are rather vulnerable to severe noise. If too many of them are scrambled or lost, the transfer can get out of synch, and the transferring programs can lose track of where they are.

One other place Kermit is vulnerable is in the beginning of a data packet. The first several bytes of these longer packets are reserved for control information: packet type, byte count, sequence number, etc. If line noise repeatedly coincides with the transmission of this control information, it is very easy for the transfer to get confused--particularly if the packet numbering gets garbled.

If you detect frequent line noise after you've connected to a host, but before you begin transfers (you will probably see extraneous junk characters appearing on your screen), you're probably in for trouble. Once transfers actually begin, line noise problems are often characterized by incrementing of the RETRY counter on the Kermit-65 transmission status display, and/or by long pauses in incrementing of the status display byte counter.

To minimize line noise, first see if there are any obvious loose connections in your equipment (telephone line connection to wall box, telephone line to modem, modem cabling to serial connector, or, if appropriate, cabling from hardwire port to serial connector). If not, you may want to hang up and redial to get another telephone connection (almost every connection is unique, and you may get a better one than you had). Many line noise problems will clear up with those simple remedies, but some may be beyond your control!

If all else fails, you may also try shortening the maximum length of your data packets (SET SEND/RECEIVE PACKET-LENGTH) to possibly lessen the effects of persistent noise.

(2c)

A Kermit transfer consists of a regular and predictable exchange of initialization, data, and, ACK/NAK packets. If something (line noise, busy computer, user error, etc.) interrupts or delays this regular exchange, there must be a way for a Kermit program on at least one end to figure out something is wrong and try to get the packet exchange back on track again.

This is usually done with a timer and retry mechanism. If a Kermit does not receive an expected packet, within its timer's time limit (a timeout), it will resend its last sent packet to try to "wake up" the other Kermit (effectively by asking it to send its last packet again). This resending is repeated ("retried") a number of times before the program assumes it cannot get things on track again. Each packet resent by Kermit-65 is counted as a RETRY on its transmission status display. If Kermit-65's retry count exceeds 20, it will try to issue an error message according to what kind of packet it was waiting for and/or it will say MAX RETRY COUNT EXCEEDED. The transfer will then be aborted.

Very frequently, timeouts are caused by unexpected delays in the remote computer, or in the network thru which you connect to it. If you know that the host machine or network is very busy, and you repeatedly have aborted transfers due to timeouts, you may be able to alleviate the problem by increasing the value of the default Kermit-65 receive timeout parameter (SET RECEIVE TIMEOUT).

Other common ways that Kermit-65 can timeout and abort are: (1) if the host machine "goes down" during a transfer, (2) if the telephone, network, or hardwire connection is completely broken during a transfer, (3) if you forgot to "start up" the host Kermit and give it a transfer command (SEND, RECEIVE, or SERVER) BEFORE giving Kermit-65 a transfer command, and (4) if (2a), (2b), (2d), or (2e) problems occur and critical initialization packets are never received.

In cases (1) and (2), you will eventually probably see a CANNOT RECEIVE DATA or MAX RETRY COUNT EXCEEDED message from Kermit-65. Cases (3) and (4) may result in a CANNOT RECEIVE INIT message.

(2d)

To do effective Kermit transfers, there must be two Kermit programs working-- one on either end of a "computer connection." In addition, the two Kermits must be able to "talk to" each other in a prescribed, standard way. Although there are specific standards for writing all Kermit programs, most of them have been written by volunteers and are in the "public domain." The protocol requirements and resultant programs are generally rather complex, and it is all too easy to inadvertently program in a subtle error in a given Kermit version. Additionally, there are many "levels of ability" of Kermit programs: some can operate in server mode, some cannot. Some can transfer binary files; some cannot, etc. Unless the Kermit programs you are using are both error free, and both have the same capabilities for the transfers you wish to perform, you are in trouble!

If there is a systematic "bug" in one of the Kermit programs, or if you are asking one Kermit to do something the other can’t do, there will usually be a problem with packet exchange; in many cases the requested transfer will not even get started. You may see a Kermit-65 error message, on the transmission status display, saying that a packet was not received, or a REMOTE MESSAGE saying a packet was unexpectedly received, or one that the command cannot be executed by the other Kermit. In some cases, you may see no explanatory error messages at all; the transfer will just "hang" and will probably eventually "timeout" and abort (MAX RETRY COUNT EXCEEDED).

(2e)

If you’ve read about category (1) errors above, you can see that there are a variety of things you can do to with files or disks to mess up a Kermit transfer. You can also wreak havoc by issuing improper or illegal commands to Kermit programs. Before trying to transfer a lot of files, or trying out a new type of transfer, be sure you understand the procedure you need to follow and the various Kermit commands that will be involved.

New Kermit users often try to command their local Kermit program (e.g., Kermit-65) to send or receive a file, without having first invoked and commanded the host Kermit.

Another common error is to issue improper commands to a remote server. For example, when VAX/VMS Kermit-32 is in server mode, and you are requesting files from it via Kermit-65 commands, you cannot use a RECEIVE command; you must instead use GET.

As with Kermit program incompatibilities, illegal or inappropriate commands will often cause a problem with packet exchange; in many cases the requested transfer or action will not even get started. You may see a Kermit-65 error message, on the transmission status display, saying a packet was not received, a REMOTE MESSAGE that a packet was unexpectedly received, or one that the command cannot be executed by the other Kermit. In some cases, you may see no explanatory error messages at all; the transfer will just "hang" and will probably eventually "timeout" and abort (MAX RETRY COUNT EXCEEDED).

Except for the fact that you will probably never note a category (2) "packet exchange" error while you are "talking to" the remote system or commanding its Kermit, and that the remedies you must employ to correct the errors will be different, recovery procedures to get your file transferred correctly will be much the same as those we described at the end of the discussion on category (1) "disk/file" errors. Make sure to read that discussion for more details than we have included below.

In short, if an error prevents a given transfer from actually beginning, you will need to correct the error and repeat all the steps that preceded it. This will be more difficult if you are transferring only one file-- having commanded the remote Kermit to SEND or RECEIVE. If you have placed the remote Kermit in server mode, and an error prevents the transfer of one file, all files transferred up to that point are probably OK, and you can usually correct the problem, and get a transfer started again without having to reCONNECT back to the host.

If you are transferring a text file, and an abort occurs in mid-transfer, some data may be salvageable in the destination file, but the best rule with any type of file is to repeat the transfer, in which case the recovery procedures in the last paragraph apply.

1.8. Customizing Kermit-65

The source code to Kermit-65 is in 6502 Assembler. It has been formatted for a cross assembler which runs on a unix 2's complement machine. Files `appxas.1` thru `appxas.3` are the cross assembler for UNIX. Get the files on a UNIX system and then look at the documentation at the start. They will easily make you a xasm for Kermit. The file `appmak.unx` is the makefile to use with the xasm to reassemble all of Kermit's parts.

Kermit-65 has been separated into two assemblies, the main routines and the com card routines for the devices shown in Table 1-1. A vector has been set up in low memory for the two assemblies to communicate. Look at the working com drivers for tips on how to incorporate your version of the com driver. some things to note: It is probably best to buffer the input from the remote and to get input characters from the remote every chance you get. Note the Microtek SV-622 driver, whenever the input is checked for a character and has a character the character is put into the buffer immediately. Also when the output is checked for ready to output, if the card is not ready to output then it is checked for a character to input. All this should help prevent losing characters.

All the routines should return with the "rts" instruction. Routines which can return a true/false indication should return with the P reg zero flag set appropriately. That is: a "beq" instruction will branch on a false indication and the "bne" will branch on a true indication. The com driver should start its routines above the main routines and tell where the end of the com driver is via location $100c. If your com driver gets too large then the bsave address would have to be changed when you are saving the binary to diskette.

address	size	module	function
1003	byte	main	This is the baud rate index as follows:
			3 - 110
			4 - 135.4
			5 - 150
			6 - 300
			7 - 600
			8 - 1200
			9 - 1800
			10 - 2400
			11 - 3600
			12 - 4800
			13 - 7200
			14 - 9600
			15 - 19200
			eg:if index is 6 then line should be 300 baud
1004	byte		unused
1005	word	driver	Address of a null terminated string.
			address should point to a capitalized
			string of the drivers id
1007	byte	main	Com slot in the form $n0 where n is slot #.
1008	byte	main	Force initialization flag when 0.
			init routine should always initialize when
			this flag is 0 & then set flag non-zero.
1009	word	main	Address of the end of Kermit main routine.
100b	byte	main	Flow control is on when high bit is set.
100c	word	driver	Address of the end of the com driver.
100e	byte	driver	Time constant-used with the 1040 rtn.
100f	word	driver	Address of the end of screen save memory(//e).
1011	byte	driver	Screen saved flag.
1020	3 bytes	driver	Jump to initialization routine.
1023	3 bytes	driver	Jump to command routine. A reg has command
			0 - hang up the line
			$0b - set baud rate
			$0c - set break on the line
			$91 - do xon on the line
			$93 - do xoff on the line
			routine returns false (P reg zero flag) if unable
1026	3 bytes	driver	Jump to check for input from the line.
			routine returns false (P reg zero flag)
			if no character on line
1029	3 bytes	driver	Jump to get input character from line.
			routine returns character in A reg
102c	3 bytes	driver	Jump to put character in A reg on line.
102f	3 bytes	driver	Jump to reset com driver.
1040	3 bytes	main	Jump to Apple ROM wait rtn. microsecs delay
			=1/2(26+27A+5A*A) where A is the accumulator
1043	3 bytes	main	Jump to routine to print null-terminated string.
			X reg contains least significant byte of address
			Y reg contains most significant byte of address
			routine does not issue a carriage return.
1046	3 bytes	main	Jump to routine to read the keyboard.
			A reg contains the character read
1049	3 bytes	main	Jump to routine to print carriage rtn & line feed.
104f	3 bytes	main	Jump to routine to set characters parity.
			A reg contains the character before and after.

Table 1-4: Communications card vector area

List of most of the prodos file types.

Num	Name	OS	Definition
$00			typeless
$01	BAD	both	BAD blocks file
$02	PCD	SOS	Pascal CoDe file
$03	PTX	SOS	Pascal TeXt file
$04	TXT	both	ASCII text file
$05	PDA	SOS	Pascal DAta file
$06	BIN	both	BINary file
$07	CHR	SOS	CHaRacter font file
$08	PIC	both	PICture file
$09	BA3	SOS	Business BASIC (SOS) program file
$0A	DA3	SOS	Business BASIC (SOS) data file
$0B	WPD	SOS	Word Processor Document
$0C		SOS	SOS system file
$0D		SOS	SOS reserved file type
$0E		SOS	SOS reserved file type
$0F	DIR	Both	subDIRectory file
$10	RPD	SOS	RPS data file
$11	RPI	SOS	RPS index file
$12		SOS	Applefile diskcard file
$13		SOS	Applefile model file
$14		SOS	Applefile report format file
$15		SOS	Screen library file
$16		SOS	SOS reserved file type
$17		SOS	SOS reserved file type
$18		SOS	SOS reserved file type
$19	ADB	ProDOS	AppleWorks Database file
$1A	AWP	ProDOS	AppleWorks WordProcessing file
$1B	ASP	ProDOS	AppleWorks Spreadsheet file
$1C-$5F			Reserved
$60-$6F		ProDOS	PC Transporter (Applied Engineering)
$60	PRE	ProDOS	ProDOS preboot driver
$61-$6A		ProDOS	Reserved
$6B	NIO	ProDOS	PC Transporter BIOS and drivers
$6C		ProDOS	Reserved
$6D	DVR	ProDOS	PC Transporter device drivers
$6E		ProDOS	Reserved
$6F	HDV	ProDOS	MSDOS HardDisk Volume
$70-$9F			Reserved
$A0	WPF	ProDOS	WordPerfect document file
$A1	MAC	ProDOS	Macrofile
$A2	HLP	ProDOS	Help File
$A3	DAT	ProDOS	Data File
$A4			Reserved
$A5	LEX	ProDOS	Spelling dictionary
$A6-$AB			Reserved

Table 1-5: PRODOS file types, part 1

Num	Name	OS	Definition
$AC	ARC	ProDOS	General Purpose Archive file
$AD-$AF			Reserved
$B0	SRC	ProDOS	ORCA/M & APW source file
$B1	OBJ	ProDOS	ORCA/M & APW object file
$B2	LIB	ProDOS	ORCA/M & APW library file
$B3	S16	ProDOS	ProDOS16 system file
$B4	RTL	ProDOS	ProDOS16 runtime library
$B5	EXE	ProDOS	APW shell command file
$B6	STR	ProDOS	ProDOS16 startup init file
$B7	TSF	ProDOS	ProDOS16 temporary init file
$B8	NDA	ProDOS	ProDOS16 new desk accessory
$B9	CDA	ProDOS	ProDOS16 classic desk accessory
$BA	TOL	ProDOS	ProDOS16 toolset file
$BB	DRV	ProDOS	ProDOS16 driver file
$BC-$BE			Reserved for ProDOS16 load file
$BF	DOC	ProDOS	document file
$C0	PNT	ProDOS	//gs paint document
$C1	SCR	ProDOS	//gs screen file
$C2-$C7			Reserved
$C8	FNT	ProDOS	Printer font file
$C9		ProDOS	finder files
$CA		ProDOS	finder icons
$CB-$DF			Reserved
$E0	LBR	ProDOS	Apple archive library file
$E1			Unknown (unlisted)
$E2	ATI	ProDOS	Appletalk init file
$E3-$EE			Reserved
$EF	PAS	ProDOS	ProDOS Pascal file
$F0	CMD	ProDOS	added command file
$F1-$F8		ProDOS	User defined filetypes (popular ones include:)
$F1	OVL	ProDOS	Overlay file
$F2	DBF	ProDOS	Database file
$F3	PAD	ProDOS	MouseWrite file
$F4	MCR	ProDOS	AE Pro macro file
$F5	ECP	ProDOS	ECP batch file
$F6	DSC	ProDOS	description file
$F7	TMP	ProDOS	temporary work file
$F8	RSX	ProDOS	linkable object module
$F9	IMG	ProDOS	ProDOS image file
$FA	INT	ProDOS	Integer BASIC program
$FB	IVR	ProDOS	Integer BASIC variables file
$FC	BAS	ProDOS	AppleSoft BASIC program
$FD	VAR	ProDOS	AppleSoft BASIC variables file
$FE	REL	ProDOS	ProDOS EDASM relocatable object module file
$FF	SYS	ProDOS	ProDOS8 system file

Table 1-6: PRODOS file types, part 2

Index

Table of Contents

List of Figures

List of Tables