MACINTOSH KERMIT USER GUIDE

For the Apple Macintosh 512, Plus, II, and SE

Christine Gianone, Frank da Cruz

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

Paul Placeway

Department of Computer and Information Science
Ohio State University, Columbus, Ohio 43210-1277

May 26, 1988

1. MACINTOSH KERMIT

Program:	Bill Catchings, Bill Schilit, Frank da Cruz (Columbia University), Davide Cervone (University of Rochester), Matthias Aebi (ECOFIN Research and Consulting, Ltd., Zuerich), Paul Placeway (Ohio State University).
Language:	C (MPW)
Documentation:	Christine Gianone, Frank da Cruz, Paul Placeway
Version:	`0.9(40)`
Date:	May 26, 1988

MacKermit Capabilities At A Glance:

Local operation:	Yes
Remote operation:	Yes (server mode only)
Login scripts:	No
Transfer text files:	Yes
Transfer binary files:	Yes
MacBinary transfers:	No
Wildcard send:	Yes (whole HFS folders)
File transfer interruption:	Yes
Filename collision avoidance:	Yes
Can time out:	Yes
8th-bit prefixing:	Yes
Repeat count prefixing:	Yes
Alternate block checks:	Yes
Terminal emulation:	Yes (VT100,VT102)
Communication settings:	Yes (Speed, Parity, Echo)
XON/XOFF:	Yes
Transmit BREAK:	Yes
IBM mainframe communication:	Yes
Transaction logging:	Yes
Session logging:	Yes
Debug logging:	No
Packet logging:	No
Act as server:	Yes
Talk to server:	Yes
Advanced server functions:	Yes
Local file management:	Yes
Command/Init files:	No
Key redefinition/macros:	Yes
File attributes packets:	No
Command macros:	No
Raw file transmit:	No
Long packets:	Yes
Sliding windows:	No

1.1. Introduction

Macintosh Kermit, or "MacKermit", is an implementation of the Kermit file transfer protocol for the Apple Macintosh family of computers. It was developed at Columbia University, based on C-Kermit (which also forms the nucleus of Unix Kermit and many other Kermit programs). Version 0.9 of MacKermit runs on the Macintosh 512, XL (Apple Lisa running MacWorks), 512e, Plus, SE, and II, under the regular Finder and the Multifinder, with which it can transfer files in the background. MacKermit 0.9 probably does not run on a 128k (original, classic) Macintosh, due to lack of sufficient memory, but should run OK on a "fat Mac" (a 128K Mac upgraded to 512K). Version 0.8 should be retained for 128K Macs.

This manual assumes you are acquainted with your Macintosh, and that you are familiar with the general ideas of data communication and Kermit file transfer. A very brief overview is given here, but for details consult the early chapters of the *Kermit User Guide* (of which this document is a chapter), or the book Kermit, A File Transfer Protocol, by Frank da Cruz, Digital Press (1987). For further information about Kermit documentation, updates, lists of current available versions, and ordering information, write to:

Kermit Distribution
Columbia University Center for Computing Activities
612 West 115th Street
New York, NY 10025 (USA)

1.2. Installation

Before you can use Macintosh Kermit or any other communication program on your Mac, you must have a way to connect it to the other computers you wish to communicate with. This means either a direct cable connection (usually using a "null modem" cable), or a modem connected to your Mac and to a telephone. The Macintosh poses two special problems at this level. First, its connectors are not the standard 25-pin RS-232 style, but either 9-pin or 8-pin special connectors which you need special adapters for. Second, the Macintosh does not supply a Data Terminal Ready (DTR) signal, which is required by most modems before they will operate. To use your Mac with a modem that is not designed specifically for the Mac, you have to either (a) configure the modem to ignore the DTR signal, or (b) feed some other active signal into the modem's DTR input. The former is usually done with DIP switches on the modem, the latter can be done in the connector that plugs into the modem by installing a jumper wire between DTR (pin 20) and DSR (pin 6), or by connecting the Mac's +12V output (pin 6 on the Mac's 9-pin connector) to DTR (pin 20) on the modem end.

If you have received a Macintosh Kermit diskette from Columbia University, there's no special software installation procedure -- just insert the diskette, double-click on the appropriate start-up file, or on MacKermit itself, and go. If all the communication and other settings agree with your requirements, there's nothing else you need to do. This process is illustrated in the next section, just below.

MacKermit is not copy-protected, and nothing out of the ordinary is required to make copies onto other diskettes, or onto your hard disk if you have one. Just use the ordinary Macintosh methods of copying files, folders, etc.

Later, you may wish to create settings files tailored to your communication environment, and you might also want to customize the keyboard configuration. Use the various Settings options for this, and then select Save Settings from the File menu. Settings and settings files are explained in Sections 1.10 and 1.11.

1.3. Getting Started

Kermit programs perform two major functions, terminal emulation and file transfer. Before transferring files between two systems you must establish a terminal connection from your system to the other one, either direct or else dialed up using a modem. Then to transfer files, login to the remote system if necessary, start up a Kermit program there, and then tell the two Kermit programs which files to transfer, and in what direction.

Most Kermit programs present you with a prompt, in response to which you type a command, repeating the process until you exit from the program. If you want to establish a terminal connection to another system, you must give the CONNECT command. Unlike these programs, MacKermit is *always* connected, and whatever keystrokes you type are sent to the other system. To give commands to MacKermit itself, you must use the mouse to pull down menus from the menu bar that overlays your terminal session, or type special Command-key equivalents.

The following example shows how to transfer a file with MacKermit. The remote computer is a Unix system, but the method is the same with most others.

- First insert the MacKermit diskette. It should appear on the screen as a diskette icon titled **Kermit**

0.9(40). Click on it twice to open if it did not open automatically when you inserted it in the drive.

- Once the disk is opened, you will see three MacKermit icons across the top of the screen. For the Unix system and most others you can use the "**Normal Settings**" icon -- to start the Kermit program click twice on it. For linemode connections to IBM mainframes, you would click twice on the "**IBM Mainframe Linemode Settings**" icon.

- You will see a white backround with menus stored under the headings **File**, **Edit**, **Settings**, **Remote**, and **Log**.

- Move the mouse pointer to the **Settings** menu and select **Communications...** by clicking on it once.

- MacKermit normally sets the communication speed to 9600 bits per second. Click on the circle in front of 1200 (or whatever speed you need to match the baud rate of your modem and/or remote system). Check to see that the other communication settings like parity are as required, and make any necessary changes.

- Click on the "`OK`" box to accept the settings.

- If you have a Hayes-like dialout modem, follow the next two steps:

    1. Type AT (uppercase) and then press the Enter key. The modem should respond with "OK" or the digit "0" (zero). If it doesn't, check the cable, the modem, etc (consult your modem manual for details).

    2. Now type ATDT 7654321 followed by Enter (replace 7654321 by the actual phone number). If the connection succeeds, you'll get a message like CONNECT (or the digit "1"), otherwise you'll see an error message like NO CARRIER, ERROR, etc, or a digit like 3 or 4 (see your modem manual).

  For non-Hayes-compatible modems, follow the instructions in your modem manual. For direct connections, skip this step altogether.

Now you should be talking to the Unix system. Type a carriage return to get its attention, and it will issue its login prompt. In the examples below, underlining is used to show the parts that you would type.

`Login: christin`	*Login to the host.*
`password:______`	*(Passwords normally don't echo.)*
`% kermit`	*Run Kermit on the host.*
`C-Kermit>receive`	*Tell it to receive a file.*

Now tell MacKermit what file to send:

- Use the mouse to point to the **File** menu and select the **Send File...** option. You can either type in the name of the file (if you know the name) or select the alternate drive to see what files are on the disk. Once you see the file you want to send, click on the filename and then click on the SEND option (or you can just click twice on the filename).

- A "File Transfer Status" box will appear to report the progress of the transer. *NOTE:* If the number of retries is increasing but not the number of packets, you should check your **Communications...** settings under the **Settings** menu.

- When the file transfer is completed, the "File Transfer Status" box should disappear and the C-Kermit prompt should reappear.

You have just transferred a file from the Macintosh to the Unix system. To transfer a file in the other direction, use the "`send` *filename*" command on Unix instead of "`receive`", and click on "**Receive File...**" from the Mac's **File** menu, instead of "**Send File...**".

After the file is transferred, your terminal connection is automatically resumed. Once your Unix session is complete, you can log out, and then exit from MacKermit:

```
C-Kermit>exit
% ^D                              Logout from Unix by typing Ctrl-D.
```

1. Select the **Quit** option in the **File** menu by clicking on it.
2. Select the **Close** option in the **File** menu by clicking on it (assuming you want to close the current folder).
3. Select the **Eject** option in the **File** menu by clicking on it (assuming you ran Kermit from a diskette that you want to eject).

That's the easiest and quickest way to use Kermit. If this simple scenario does not work for you, look for any obvious incorrect settings (speed, parity), fix them, and try again. Otherwise, read on.

1.4. The Macintosh File System

The Macintosh file system consists of one or more disks, each disk containing files. There are actually two Macintosh file systems, which work slightly differently.

Disks formatted for the older Macintosh File System (MFS) are essentially "flat". All files on one of these disks must have a unique name. Files may be collected together into "folders", but folders are not analogous to directories on other file systems, and no two folders on the same disk may contain files of the same name; the folders exist only to make things look nicer in the Finder. All Macintoshes have support for MFS.

Disks formatted with the newer Hierarchical File System (HFS) are not "flat"; each folder is a directory. There may not be more than one file with the same name in a single folder, but there may be identically named files in different folders.

Macintosh file names may contain practically any printable characters, including space and punctuation -- but colon (‘‘`:`’’) may not be used; it is used in device names and as the HFS path element separator.

1.5. Menus

The major menus are **Apple**, **File**, **Edit**, **Settings**, **Remote**, and **Log**. The **Apple** menu gives some information about the program, including the MacKermit version number and the C-Kermit protocol mudule version number (useful in reporting bugs). It also shows statistics about the most recent file transfer.

The **File** menu invokes Kermit's file transfer functions, **Send**, **Get**, and **Receive**. It also allows settings to be saved and restored, and like most Macintosh applications, includes a "quit" entry for leaving the program, and a "transfer" entry for transferring to another program without going through the Finder.

The **Edit** menu provides support for Macintosh desk accessories that need to have this menu to do cut and paste. This menu does not currently do anything in MacKermit.

The **Settings** menu provides dialog boxes for file, communications, and protocol settings; these will be discussed below.

The **Remote** menu has the commands that can be sent to Kermit servers, as well as an option to turn Macintosh Kermit itself into a server (also discussed below).

The **Log** menu contains commands to start and stop session and transaction logging. It also has an entry to dump the current screen image to the session log, which is only enabled when the session log is open.

1.6. Terminal Emulation

Before you can transfer files, you have to establish a terminal connection with the other computer. You don't have to give MacKermit any special command to do this, just start the program. Assuming you have a physical connection, then the software will use it. If you think you have a physical connection, but don't see any results, click on the **Settings** menu and select **Communications** to make sure you have the right speed and parity. If you have to dial out to make the connection, you must do this yourself -- Mac Kermit won't do it for you. Depending on the type of modem, you must either type dialing commands to it directly (like the Hayes ATDT command in the example in section 1.3), or else dial the phone manually, wait for an answer and a carrier tone, and then put the modem in data mode.

Once you've made the connection, you can use MacKermit's terminal emulator, which conforms to ANSI standard X3.64, providing a subset of the features of the DEC VT102 terminal (a VT100 with line and character insert and delete functions added). The functions provided are sufficient to allow MacKermit to act as a terminal for the EMACS full-screen editor as it exists on most timesharing systems, and for most host-resident display-oriented applications that expect to do cursor positioning and editing on the VT100 or VT102 screen, such as VAX TPU. MacKermit does not currently support the following VT100/102 functions:

- Double height or double width lines
- Blinking
- 132 columns
- DEC-style line wrapping
- Control characters embedded in escape sequences
- VT52 mode

(this is not an exhaustive list)

The keyboard is set up by default as follows: If your Macintosh has a Control key (ie. an SE or II), Kermit uses it, and the Command (Fan, Cloverleaf) key can be used for keyboard equivalents for menus. If your Mac does not have a Control key, then the Command key is used as the Control key. The CAPS LOCK key forces all alphabetic characters to upper case. The terminal emulator sends ESC (escape) when the ‘‘`’’ (accent grave) key is pressed unshifted (even if your keyboard has an ESC key). The character ‘‘`’’ can be sent by typing Control (or Command) and the same key. The Backspace key sends a Delete (Rubout) and Control-Backspace sends a Backspace. On the original Mac keyboards, the main keypad Enter key sends a "short" (250ms) BREAK signal. The Mac+, Mac SE, and Mac II do not have a main keypad Enter key, so the BREAK function must be reassigned to another key.

You can modify the keyboard layout any way you like, defining keyboard macros, defining or moving the Control and Escape keys, etc., using MacKermit's built-in key configuration features. Older MacKermits (version 0.8 and earlier) came with a separate key configuration program called CKMKEY. This should not be used, because it does not understand the format of the 0.9 and later keyboard configuration software.

MacKermit includes a mouse-controlled cursor positioning feature for use during terminal emulation. If the "Mouse -> Arrow Keys" feature is turned on (via the **Terminal** entry of the **Settings** menu), then when the mouse button is pressed, the program acts as if you typed the VT100 keypad arrow keys to move the terminal cursor to where the mouse cursor is. MacKermit does this by sending the absolute strings for arrow keys, independant of what is bound to the actual arrow keys of the keyboard.

MacKermit sets the Mac hardware to do 8-bit data communication with no parity, and then correctly sets the parity bit of each character itself in software, as requested in the **Communication** settings menu. This has the benefit of avoiding the problem of a machine which requires a different input parity than it sends back. MacKermit will correctly receive all of the characters sent to it, no matter which parity they are.

To allow useful coexistence of desk accessories and Kermit, the terminal emulation window may be dragged using the drag bar. If a desk accessory overlays the emulation window, the emulation window can be clicked upon to move it in front of the DA, and later dragged to reveal the hidden desk accessory so that it can be restored to the foreground. The same thing can be done with Kermit's own remote response window as well. Note that Kermit's terminal emulation window does not accept input when any other window is in the foreground.

MacKermit uses XON/XOFF (control-Q and control-S) flow control during terminal emulation and file transfer. If the other system does not understand XON/XOFF, problems may result at high speeds. The terminal emulator can normally keep up at 9600 baud, and has a very large input buffer, but after several continuous scrolling screens at this speed, some characters may be lost. When running at high baud rates on a system that does not understand XON/XOFF flow control, either keep your terminal in page mode, use a text paging program such as Unix "more", or view text with a non-scrolling screen editor. Also, don't drag the terminal emulation window while characters are arriving; if you do, the characters may be lost and the display may become confused.

During terminal emulation, the characters displayed on the screen may also be saved on the disk. This allows you to record interactions with the remote system, or to "capture" files that you can't transfer with Kermit protocol, for example when the remote system does not have a Kermit program. Use the **Log** menu, and choose session logging to activate this feature. The result goes into a file called "Kermit Session" in the current folder, which is always appended to, rather than overwritten. To create a new session log, delete or rename the old one first.

The following features are missing from the MacKermit terminal emulator, and may be added in subsequent releases:

- Restoration of character attributes such as underlining or highlighting.
- Cutting text from screen to clipboard.
- Transmission of raw text to host (e.g. pasting to screen).
- Screen rollback.
- Screen resizing.
- Explicit modem or dialer control.
- Login scripts.
- Printer support.
- Ability to use the printer port for terminal emulation.
- A way to disable XON/XOFF flow control, or select other flow controls.

1.7. File Transfer

Like most Kermit programs, MacKermit allows you to send and receive text or binary files singly or in groups. It will interact with a remote Kermit server, and it can act as a server itself. However, due to the unique nature of the Macintosh file system, there are some special considerations:

- <u>Mode</u> - Text or Binary. Binary means the data is sent or stored without modification. Text means that every carriage return character (CR) in a Macintosh file is translated to a carriage-return-linefeed (CRLF) sequence when sending, and every CRLF in an incoming file is turned into a CR when stored on the Mac disk. A text file is produced when you save a file from MacWrite or other applications using the "text only" option; text files are not associated with any particular Macintosh application and can be sent in a useful fashion to other kinds of computers.

  A word of caution about Macintosh text files: The Macintosh supports an extended version of ASCII, with characters like accented and umlauted vowels in the 128-255 range. These characters allow representation of Roman-based languages other than English, but they do not follow any of the ISO standards for extended character sets, and thus are only useful on a Mac. When transferring text files, you should ensure that either there are no extended characters in the file, or that the other system can understand the Mac's 8-bit characters.

- <u>Fork</u> - Data or Resource. Macintosh files may have two "forks". The data fork contains data for an application; the resource fork contains icons, strings, dialog boxes, and so forth. For instance, a MacWrite document contains text and formatting information in the data fork, and fonts in the resource fork. For applications, the executable code is stored in the resource fork.

File transfer is initiated when you select **Send file...**, **Receive File...**, or **Get file from server...** from MacKermit's **File** menu.

File transfers can be canceled by clicking on the Cancel File or Cancel Group buttons. These will always work when sending. When receiving, they will work if the opposite Kermit honors this (optional) feature of the protocol. There is also an "emergency exit" from any protocol operation, which can be taken at any time by typing "Command-." -- that is, hold down the Command (Fan, Cloverleaf) key and type period.

The progress of file transfer operations can be logged into a Macintosh file called a "transaction log". This log will show the names of the files transferred, the date and time, and the completion status. This feature is useful with long unattended transfers -- you can come back later and read the transaction log to find out what happened. The transaction log is called "Kermit Log".

The current version of Mac Kermit can only send one fork of a file at a time. When a file has two forks, there is no provision for sending both forks together. This restriction may be lifted in future releases of MacKermit, for example by converting applications to MacBinary format during transmission.

1.7.1. Sending Files

To send files, first put the remote Kermit in server mode, or else give it the RECEIVE command. Then use the mouse to select **Send file...** from the **File** menu. This will give you a MacKermit file-open box, which includes the standard Macintosh dialog items -- a file list, Disk and Eject buttons, etc. You can either send one file at a time, by clicking on its name in the file list, or send the entire contents of the current HFS folder (for HFS disks only, of course). Clicking the Disk button will switch the file list to another physical disk. If desired, you can type an alternate name to send the file under. When you select a file, MacKermit examines its type; if the type is APPL, then MacKermit expects to send the resource fork in binary mode, otherwise the data fork in text mode. The Mode and Fork radio buttons will display these choices; you may change them before clicking the Send button.

1.7.2. Receiving Files

You can receive or get multiple files, providing the opposite Kermit is capable of sending multiple files in a single transaction (most are). To receive files, first give the remote Kermit a SEND command and then select **Receive file...** from the **File** menu. To get files from a server, first put the remote Kermit into server mode, then select the **Get file from server...** option from the **File menu**, and type in the name of the file you want to get, or a wildcard designator for multiple files, in the remote system's filename syntax.

As each file arrives at the Mac, it will be decoded according to the current mode (text or binary), and stored in the default fork (data or resource). The file names will be either the names the files arrive with (overwriting existing files of the same names) or new unique names (when name conflicts occur), according to the current default for name collisions. You may also elect to perform an "attended" receive, in which you have an opportunity to override file defaults on a per-file basis (do this in the **Protocol** section of the **Settings** menu). But attended operation must be used with caution -- if you take too long (more than about a minute) to execute an incoming file's dialog box, the opposite Kermit could time out and terminate the transaction. If this happens, tell the opposite Kermit to send again and try again with the receive dialog.

The folder for new files is the same as the location of the settings file, or if no settings file was used then the new files appear on the desktop. If you are transferring a lot of files and want to keep them together, create a folder, drag the settings file into it, and double click on the settings file; all created files will appear in that folder.

1.8. Remote Commands

When connected to a Kermit server, MacKermit is capable of issuing special file management and other commands to it. The **Remote** menu contains these commands. You may request directory listings, you can delete files, change directories, etc, on server's machine. The response from these commands (if any) is displayed in a special pop-up window. Responses to multiple Remote commands are separated by a dashed line. The response window can be scrolled, sized, and positioned, and can be hidden by clicking the menu item "**Hide Response**" or the window's go-away box; all text remains intact and will be appended to the next time you do a Remote command; it can also be brought to the foreground by clicking the **Show Response** menu item. Note that typein to the terminal emulator will not take effect when the response window -- or any other window (such as a desk accessory) -- is up front. This is not a bug, but a feature of the Macintosh user interface guidelines.

If the response buffer gets too full (greater than 30,000 characters), MacKermit will remove enough text from the beginning of the buffer, in 512 byte chunks, to make it less than 30,000 characters again.

A Remote command can be canceled by taking the Emergency Exit (Command-`.`). To disengage from the remote Kermit server, click on **Bye** or **Finish** in the **Remote** menu.

1.9. Server Operation

MacKermit may itself act as a Kermit server. Just set the desired parameters in the **Settings** menu, then click on **Be a Server** in the **Remote** menu. The MacKermit server can respond to SEND, GET, REMOTE DIRECTORY, FINISH, and BYE commands. You can send single or multiple files to a MacKermit server, and you can get a single file from it by name. You can also get all the files in the current folder by using a colon (‘‘`:`’’) as the file specification in the GET command:

```
GET :
```

If you give the FINISH command, MacKermit will return to terminal mode. If you give the BYE command, the Macintosh will reboot itself.

You can take MacKermit out of server mode from the Mac keyboard by typing the emergency exit sequence, Command-dot.

1.10. Settings

You can change File, Communications, Protocol, Terminal, Keyboard macros, and Keyboard modifier settings by using the **Settings** pull-down menu. You can save and load these settings by invoking the appropriate selection in the **File** menu. If the "bundle bit" has been correctly set on your version of MacKermit (it should be), then you can double-click on the resulting document to start MacKermit with those settings.

The **File** settings establish the defaults for file transfer:

- <u>Attended</u> versus <u>Unattended</u> operation for incoming files.
- <u>Naming</u>: When doing <u>unattended</u> file reception, whether incoming files should supersede existing files of the same name, or a new unique name should be assigned to them. If the latter, the new name is formed by adding a dot and a number to the end. For instance, if a file called FOO exists and a file called FOO arrives, MacKermit will store the arriving file as FOO.1; if FOO.1 exists, then FOO.2, etc.
- <u>Mode</u>: text or binary. Used for received files only. When sending, MacKermit tries to figure out an appropriate mode for the file being sent (but then lets you override it the Send File dialog).
- <u>Fork</u>: which fork -- data or resource -- to send, or to store an incoming file into.

The **Communications** settings allow you to set the baud rate (anywhere between 300 baud and 57.6K baud, except 38.4K baud), and parity (odd, even, mark, space, or none). When the parity is set to none the Macintosh uses an 8-bit-wide connection. All other parity settings tell the Macintosh to use a 7-bit-wide connection, and to request 8th-bit prefixing when transferring 8-bit data. If the remote host or the communication path uses any kind of parity, then you won't be able to transfer files successfully unless you tell MacKermit (and in most cases also the Kermit on the other end) about it. Duplex is selected in the **Terminal** settings.

The **Protocol** settings allow you to set packet parameters for both incoming and outbound packets. These include the block check type (1 or 2 character checksum, 3-character 16-bit CRC-CCITT), line turnaround handshake character (for file transfer with half duplex systems), packet start and end characters, padding, packet length, timeout interval, and packet length. Characters are specified by entering their ASCII value in decimal, e.g. 1 for Control-A, 13 for Control-M (Carriage Return), etc. The RECEIVE parameters are conveyed by MacKermit to the other Kermit. For instance, if you set the receive-packet-length to 500, MacKermit will tell the other Kermit to send 500-character packets. The SEND parameters are used to override negotiated values, and need rarely be used.

Long packets are selected by setting the RECEIVING packet length between 95 and 1000. Normally, you should not change the sending length because MacKermit, and most other Kermits, will configure themselves correctly. Note also that the fastest file transfers will happen with long packets in the range of 300-500. Very long packets actually end up being much slower, because the operating systems in both the Mac and the other machine have to do more work to cope with such long inputs, and, under noisy conditions, the probability is higher that a longer packet will be struck by noise, and will take longer to retransmit.

The **Terminal** settings let you modify the characteristics of the VT102 emulator, such as auto-linefeed, autowrap, autorepeat keys, block vs underline cursor, blinking vs steady cursor, inverted screen (reverse video), and smooth scrolling. There is also a "visible bell" for those who can't hear the audible bell produced upon receipt of a Control-G, and an option to display control characters visibly by showing their numeric ASCII values (in decimal) in a single character cell. If local echo is needed, as in half-duplex connections, that must be specified here also.

1.11. Settings Files

You can start MacKermit with all its "factory settings" by double clicking on the MacKermit icon. Factory settings are designed for direct communication with most other microcomputers, DEC minis and mainframes, etc: 9600 bps, no parity, XON/XOFF, remote echo, etc. You can change the communication, protocol, file, keyboard, and terminal settings by going through the options in the **Settings** menu. Once you have set all parameters as desired, you can save your settings in a "MacKermit settings file" by selected "**Save Settings...**" from the **File** menu. A settings file is, in Macintosh terminology, a "MacKermit document". You'll recognize it because it looks like a dog-eared piece of paper with the MacKermit icon superimposed. You can have more than one settings file.

There are two ways to use a settings file. First, you can double-click on it, just as you can double-click on a MacWrite document to start up MacWrite to edit a particular file. This method starts up MacKermit with all the saved settings. The other method is to click on the "**Load Settings...**" option in the **File** menu from inside MacKermit. This lets you change settings without leaving and restarting the program. **Load Settings...** shows all MacKermit settings files in the selected folder. Opening one of them loads all its settings, removing all current settings.

You can "edit" a MacKermit settings file by loading it, going through the `Settings` menu, and then saving the settings either in a new file, or overwriting the same file.

As distributed by Columbia, Mac Kermit comes with two settings files. One is called "Normal Settings", and is pretty much identical to Mac Kermit's factory settings. The other is "IBM Mainframe Linemode Settings". It selects mark parity, local echo, XON half-duplex line turnaround handshake. You can use these files as-is, customize them for your own environment, or create new settings files for all the different kinds of systems that you use.

1.12. Reconfiguring the Keyboard

Beginning with version 0.9, MacKermit has keyboard configuration functions built in. These are accessed through the **Set Key Macros** and the **Set Modifiers** entries in the **Settings** menu.

The Macintosh keyboard is composed of normal keys and modifier keys. Modifier keys are those keys that, when held down, change the meaning of other keys. On the Mac these are: SHIFT, CAPS LOCK, OPTION, CONTROL (only on the Mac II and SE), and COMMAND (also known as APPLE, CLOVER, or FAN). Normal keys are the letters, numbers, special symbols, arrow keys, space bar, and function keys. Only one normal key can be typed at a time, but one or more modifier keys can be pressed down along with it.

When you type a key, Kermit reads both the ASCII value, and the keyboard-independent scan code for that key. Kermit looks in its table of key macros to see if there is a macro for this combination of key and modifiers, and if so sends the macro. If there is no macro, Kermit then looks in its modifier table to see if any of the modifiers do special things to the character; if so, it does these to the character. Finally, Kermit sends the character. In the normal case when there is no macro and no modifiers apply, the character sent is simply the ASCII value for that character.

It is important to keep in mind that if the parity setting is something other than none, the high (8th) bit will be stripped off of the characters when they are transmitted. Since most systems do not understand characters in the range 128 -- 255 (decimal), you should avoid using the Apple extended characters (accented vowels, for example) during terminal connection.

1.12.1. Defining Key Macros

To define a new key macro, select the **Key Macros** entry. A dialog window will appear, asking you to press the key to define. Type the key (including any of the modifiers). A new dialog will appear, with an editable text field in it. Enter the definition for the key here. Your definition may be up to 255 characters long, and can include all of the control characters (including NUL). Special characters can be included in the macro by entering a ‘‘`\`’’ (backslash), followed by up to 3 *octal* (base 8) digits for the value (just like in the C programming language). For example, an ASCII NUL (value 0) would be written as ‘‘`\000`’’, carriage return (ASCII 13) would be written ‘‘`\015`’’ (1 x 8 + 5 = 13). Also, control characters may be entered with a backslash, followed by a caret (or circumflex, ‘‘`^`’’), followed by the corresponding letter. Thus a Control-G (value 7) could be entered as ‘‘`\007`’’, ‘‘`\^G`’’, or ‘‘`\^g`’’. To include a literal backslash in a definition, type in two of them: ‘‘`\\`’’.

BREAK conditions are also programmable as macros. If the entire macro the string is ‘‘`\break`’’, then typing the defined key will send a short (1/4 second) break. A long (3.5 second) BREAK is defined with ‘‘`\longbreak`’’. Note that a macro can define either a BREAK, or a string of normal characters, but not both.

1.12.2. Defining Key Modifiers

Skip ahead to the next section if you already know about things like SHIFT, CAPS LOCK, CONTROL, and META.

On a typewriter the only modifier key is SHIFT. Typing a character with no modifier key depressed selects a lowercase letter or the character printed on the lower face of the keytop (say, the digit "4"). Typing a character with SHIFT depressed selects an uppercase letter or the character printed on the upper face of the keytop (say, a dollar sign). Some keyboards also have a SHIFT LOCK key, which stays down once pressed and pops up the next time it's pressed; its operation is equivalent to holding down SHIFT. And some keyboards have a CAPS LOCK key which operates like SHIFT LOCK, but only upon letters.

Computer terminals also have a modifier key called CONTROL (or CTRL). Its function is a little less obvious: it is intended to produce one of the 33 characters in the "control range" of the ASCII alphabet. Control characters are not graphic -- they are intended for use as format effectors (like carriage return, formfeed, tab, backspace), for transmission control, or for device control. The remaining 95 characters -- letters, digits, punctuation, and space --

are the graphic characters. When a character is typed with the CONTROL modifier pressed, its "control equivalent" (if any) is transmitted. By convention, the control equivalent of A is Control-A, B is Control-B, etc, and there are also seven special control characters generally associated with punctuation characters or special keys. For the "alphabetic" control characters Control-A through Control-Z, SHIFT or CAPS LOCK modifiers are ignored; for the others, operation varies from terminal to terminal.

The SHIFT and CONTROL modifiers allow all 128 ASCII characters to be sent from a normal typewriter-like keyboard that has about 50 keys. However, certain host-resident computer applications -- notably the full screen text editor EMACS and its descendents -- can be used to greater advantage with a 256 character 8-bit alphabet (EMACS responds to single-character commands, and the more characters a terminal can send, the more commands are directly available).

For this purpose, some terminals also provide a META modifier key. This key simply causes the high-order ("8th") bit of the selected 7-bit ASCII value to be set to 1 upon transmission. This can only work when the connection is 8-data-bits-no-parity. When parity is in use, EMACS allows a sequence of two 7-bit ASCII characters to represent a single meta character. The advantage of having a real META modifier key is that it can be held down while the actual key is struck repeatedly or even autorepeats, whereas a use of a "meta prefix" such as <escape> requires much more typing. To illustrate, suppose META-F is the command to go forward one word. If you want to execute this operation repeatedly, just hold down META and F and let it autorepeat. If you don't have a META key, then you'd have to type <escape>F<escape>F<escape>F..., etc.

A common problem faced by computer users who switch from one terminal or PC to another is the placement of the modifiers and other special keys. DEC, IBM, Apple, and other manufacturers consistently move these keys around on new models of their keyboards. MacKermit allows you to assign any of various functions to any of the Mac's modifier keys, and to assign any desired character or character sequence to the regular keys, so that you can tailor the layout of your Mac's keyboard to suit your taste.

1.12.3. Modifiers Dialog

To change the action of any of the modifier keys, select **Modifiers** from the **Settings** menu. A dialog will appear that looks roughly like the one in Figure 1-1 (the ‘‘%’’ represents the Apple or Clover key).

```
   Modifier Pattern:  -->   Modification:

Ctrl Opt Lock Shift  %      Unmodify Caps Ctrl Meta Prefix string:

[X]  [ ] [ ]  [ ]   [ ]  |   [ ]     [ ]  [X]  [ ]  [____________]
[ ]  [ ] [ ]  [ ]   [X]  |   [ ]     [ ]  [X]  [ ]  [____________]
[ ]  [X] [ ]  [ ]   [ ]  |   [x]     [ ]  [ ]  [ ]  [\033________]
[ ]  [ ] [ ]  [ ]   [ ]  |   [ ]     [ ]  [ ]  [ ]  [____________]

              (Cancel)   (Help)   ( OK )
```

Figure 1-1: MacKermit Key Modifier Dialog

The check boxes are divided into rows, each one describing a modification. The left half of each row describes the modifier combination to look for; a checked box means that this key is down, and an unchecked box means "don't care". Note that there is no way to specify a key being up, and lines with nothing checked on the left side will be ignored; the character will be modified in the normal Macintosh way.

The right half describes what modification to do to the characters. The Unmodify modification says "make this the character that would be sent from the same key with no modifer keys pressed". In other words, un-Option, un-Caps, un-Control, and un-Shift this character. The Caps modification translates all letters to upper case, Ctrl makes the letter a contol character, Meta sets the high (8th) bit on the character, and if a Prefix string is present, it is sent before the character is.

Hints about modifiers:

- Beware of the Option key. It changes the value of any characters you use with it. If you type Option-F, the Mac will send a D, if you type Option-B, the Mac will send a ‘‘`:`’’, etc. If you want to use the option key as a modifier, be sure to check the "Unmodify" box.
- To use MacKermit with a version of EMACS that does not accept 8-bit Meta characters, define a key, like Option, to be unmodified, with a prefix string of `\033` (ASCII Escape), as in Figure 1-1. Then you can hold down Option and type F (or any other key) repeatedly, or let it autorepeat, and MacKermit will send the correct prefix-Meta sequence.
- When interpreting a keystoke, MacKermit checks the list of modifiers from top to bottom, applying the first one that matches. This means that if you want a different modifier for Command-Option and just plain Command, you must put the definition for Command-Option first in the list.

1.13. Bootstrapping

This section applies if you do not have a MacKermit diskette, but MacKermit is available for downloading from some other computer.

MacKermit is distributed in source form for building on a Macintosh, running Apple’s Macintosh Programmers Workbench (in MPW C), in `.HQX` "BinHex 4" form, and sometimes also as a binary resource file. Those who want to work from the source are referred to the file `CKMKER.BLD` for instructions.

If you’re downloading, it’s best to work with `CKMKER.HQX`, a textual encoding of the MacKermit application. Download this using any technique available to you -- an old release of Kermit, an Xmodem implementation, even raw screen capture. Then run BinHex (version 4) to convert it into a working application (select **Upload -> Application** from the **File** menu). Eveything will be set up correctly -- icons, forks, etc.

If you don’t have the `.HQX` file available, but you do have access to the binary resource file (its name will be `CKMKER.RSRC`, `ckmker.rsrc`, `CKMKER.RSR`, `ckmker.rsr`, `%ckmker` or some variation on these, depending on what system it’s stored on and how it got there), AND if you have "MacPut" on your system and MacTerminal on your Mac, AND if you have an 8-bit-wide (no parity) data path between your Mac and your system, then you can use MacPut to download the binary resource file to your Mac using MacTerminal’s "MacBinary" format (a variant of XMODEM). After doing this you must use a program such as SetFile or ResEdit on the Mac to set the author to KR09, the type to APPL, and turn on the bundle bit. Do not bother with the CKMKEY program, as it is not used with newer MacKermits. If you have an earlier release of MacKermit, you may use it in place of MacTerminal and MacPut.

1.14. Differences Between Versions 0.8 and 0.9

MacKermit 0.8(34) runs on the 128K Mac, the 512K Mac, and the Mac Plus, but not on the Macintosh II or SE. MacKermit 0.9(40) runs on all Macs except the 128K original. You should use version 0.9 unless you have a 128K Mac.

The second major difference is that the program is has been translated into Apple MPW C, so that it can be edited, compiled, and built on the Macintosh itself. This was done originally by Jim Noble of Planning Research Corporation, who converted MacKermit from SUMACC C (which had to be cross compiled on a UNIX system) to Megamax C. Jim’s version was converted to MPW C by Matthias Aebi, who also added most of the new features listed below. Paul Placeway integrated the program with the current (long packet) version of C-Kermit and added additional new features.

Besides these important differences, there were many other changes from version 0.8 to version 0.9, including:

- The Cursor with open desk accessories now works correctly
- Long packet support
- New program icon
- New settings files are no longer TEXT
- Settings can now be written back to an already existing settings file
- Key redefinition function built in to Kermit, no more CKMKEY
- Server mode directory listing feature
- Multifile (folder) send
- Server "Delete" file command
- Server "Space" command
- Get whole folder content from the server with filename ‘‘:’’
- Recognition of all the different Mac keyboards
- Support of menu command keys (key macros)
- Terminal settings dialog separated from communication settings
- Non-transparent terminal mode
- Display of statistics and protocol version to "About Kermit" dialog.
- Parity problems fixed
- Session logging
- Transaction logging
- Multifinder support
- Additions to the VT102 emulator (smooth scrolling, etc)
- Rearrangement of menus and displays
- Program no longer hangs if remote response window gets too full
- Program now works correctly on 64K ROM machines
- A new manual

This manual applies in large part to version 0.8(34), except that the older version is missing the new features listed above, and it comes in two pieces: CKMKER and CKMKEY. The CKMKEY program is used to program the keys, like the **Set Key Macros...** and **Set Modifiers** described in this manual, and creates a settings file which Kermit itself uses. The old version only works well with early Macintosh keyboards.

Appendix I
The ASCII Character Set

ASCII Code (ANSI X3.4-1968)

There are 128 characters in the ASCII (American national Standard Code for Information Interchange) "alphabet". The characters are listed in order of ASCII value; the columns are labeled as follows:

Bit	Even parity bit for ASCII character.
ASCII Dec	Decimal (base 10) representation.
ASCII Oct	Octal (base 8) representation.
ASCII Hex	Hexadecimal (base 16) representation.
EBCDIC Hex	EBCDIC hexadecimal equivalent for Kermit translate tables.
Char	Name or graphical representation of character.
Remark	Description of character.

The first group consists of nonprintable 'control' characters:

	.....ASCII....			EBCDIC		
Bit	Dec	Oct	Hex	Hex	Char	Remarks
0	000	000	00	00	NUL	^@, Null, Idle
1	001	001	01	01	SOH	^A, Start of heading
1	002	002	02	02	STX	^B, Start of text
0	003	003	03	03	ETX	^C, End of text
1	004	004	04	37	EOT	^D, End of transmission
0	005	005	05	2D	ENQ	^E, Enquiry
0	006	006	06	2E	ACK	^F, Acknowledge
1	007	007	07	2F	BEL	^G, Bell, beep, or fleep
1	008	010	08	16	BS	^H, Backspace
0	009	011	09	05	HT	^I, Horizontal tab
0	010	012	0A	25	LF	^J, Line feed
1	011	013	0B	0B	VT	^K, Vertical tab
0	012	014	0C	0C	FF	^L, Form feed (top of page)
1	013	015	0D	0D	CR	^M, Carriage return
1	014	016	0E	0E	SO	^N, Shift out
0	015	017	0F	0F	SI	^O, Shift in
1	016	020	10	10	DLE	^P, Data link escape
0	017	021	11	11	DC1	^Q, Device control 1, XON
0	018	022	12	12	DC2	^R, Device control 2
1	019	023	13	13	DC3	^S, Device control 3, XOFF
0	020	024	14	3C	DC4	^T, Device control 4
1	021	025	15	3D	NAK	^U, Negative acknowledge
1	022	026	16	32	SYN	^V, Synchronous idle
0	023	027	17	26	ETB	^W, End of transmission block
0	024	030	18	18	CAN	^X, Cancel
1	025	031	19	19	EM	^Y, End of medium
1	026	032	1A	3F	SUB	^Z, Substitute
0	027	033	1B	27	ESC	^[, Escape, prefix, altmode
1	028	034	1C	1C	FS	^\, File separator
0	029	035	1D	1D	GS	^], Group separator
0	030	036	1E	1E	RS	^^, Record separator
1	031	037	1F	1F	US	^_, Unit separator

The last four are usually associated with the control version of backslash, right square bracket, uparrow (or circumflex), and underscore, respectively, but some terminals do not transmit these control characters.

The following characters are printable:

First, some punctuation characters.

	.....ASCII....			EBCDIC		
Bit	Dec	Oct	Hex	Hex	Char	Remarks
1	032	040	20	40	SP	Space, blank
0	033	041	21	5A	!	Exclamation mark
0	034	042	22	7F	"	Doublequote
1	035	043	23	7B	#	Number sign, pound sign
0	036	044	24	5B	$	Dollar sign
1	037	045	25	6C	%	Percent sign
1	038	046	26	50	&	Ampersand
0	039	047	27	7D	’	Apostrophe, accent acute
0	040	050	28	4D	(	Left parenthesis
1	041	051	29	5D	)	Right parenthesis
1	042	052	2A	5C	*	Asterisk, star
0	043	053	2B	4E	+	Plus sign
1	044	054	2C	6B	,	Comma
0	045	055	2D	60	-	Dash, hyphen, minus sign
0	046	056	2E	4B	.	Period, dot
1	047	057	2F	61	/	Slash

Numeric characters:

	.....ASCII....			EBCDIC		
Bit	Dec	Oct	Hex	Hex	Char	Remarks
0	048	060	30	F0	0	Zero
1	049	061	31	F1	1	One
1	050	062	32	F2	2	Two
0	051	063	33	F3	3	Three
1	052	064	34	F4	4	Four
0	053	065	35	F5	5	Five
0	054	066	36	F6	6	Six
1	055	067	37	F7	7	Seven
1	056	070	38	F8	8	Eight
0	057	071	39	F9	9	Nine

More punctuation characters:

	.....ASCII....			EBCDIC		
Bit	Dec	Oct	Hex	Hex	Char	Remarks
0	058	072	3A	7A	:	Colon
1	059	073	3B	5E	;	Semicolon
0	060	074	3C	4C	<	Left angle bracket
1	061	075	3D	7E	=	Equal sign
1	062	076	3E	6E	>	Right angle bracket
0	063	077	3F	6F	?	Question mark
1	064	100	40	7C	@	"At" sign

Upper-case alphabetic characters (letters):

	.....ASCII....			EBCDIC		
Bit	Dec	Oct	Hex	Hex	Char	Remarks
0	065	101	41	C1	A	
0	066	102	42	C2	B	
1	067	103	43	C3	C	
0	068	104	44	C4	D	
1	069	105	45	C5	E	
1	070	106	46	C6	F	
0	071	107	47	C7	G	
0	072	110	48	C8	H	
1	073	111	49	C9	I	
1	074	112	4A	D1	J	
0	075	113	4B	D2	K	
1	076	114	4C	D3	L	
0	077	115	4D	D4	M	
0	078	116	4E	D5	N	
1	079	117	4F	D6	O	
0	080	120	50	D7	P	
1	081	121	51	D8	Q	
1	082	122	52	D9	R	
0	083	123	53	E2	S	
1	084	124	54	E3	T	
0	085	125	55	E4	U	
0	086	126	56	E5	V	
1	087	127	57	E6	W	
1	088	130	58	E7	X	
0	089	131	59	E8	Y	
0	090	132	5A	E9	Z	

More punctuation characters:

	.....ASCII....			EBCDIC		
Bit	Dec	Oct	Hex	Hex	Char	Remarks
1	091	133	5B	AD	[	Left square bracket
0	092	134	5C	E0	\	Backslash
1	093	135	5D	BD	]	Right square bracket
1	094	136	5E	5F	^	Circumflex, up arrow
0	095	137	5F	6D	_	Underscore, left arrow
0	096	140	60	79	‘	Accent grave

Lower-case alphabetic characters (letters):

	.....ASCII....			EBCDIC		
Bit	Dec	Oct	Hex	Hex	Char	Remarks
1	097	141	61	81	a	
1	098	142	62	82	b	
0	099	143	63	83	c	
1	100	144	64	84	d	
0	101	145	65	85	e	
0	102	146	66	86	f	
1	103	147	67	87	g	
1	104	150	68	88	h	
0	105	151	69	89	i	
0	106	152	6A	91	j	
1	107	153	6B	92	k	
0	108	154	6C	93	l	
1	109	155	6D	94	m	
1	110	156	6E	95	n	
0	111	157	6F	96	o	
1	112	160	70	97	p	
0	113	161	71	98	q	
0	114	162	72	99	r	
1	115	163	73	A2	s	
0	116	164	74	A3	t	
1	117	165	75	A4	u	
1	118	166	76	A5	v	
0	119	167	77	A6	w	
0	120	170	78	A7	x	
1	121	171	79	A8	y	
1	122	172	7A	A9	z	

More punctuation characters:

	.....ASCII....			EBCDIC		
Bit	Dec	Oct	Hex	Hex	Char	Remarks
0	123	173	7B	C0	{	Left brace (curly bracket)
1	124	174	7C	4F	|	Vertical bar
0	125	175	7D	D0	}	Right brace (curly bracket)
0	126	176	7E	A1	~	Tilde

Finally, one more nonprintable character:

0	127	177	7F	07	DEL	Delete, rubout

Index

Table of Contents

List of Figures