The following is a first alpha darft of the new MacKermit documentation. There are still many features not documented and some that need to be updated. Whereever you find '*****' in the text it means that there is something to be changed / filled in.

MACINTOSH KERMIT

Program: Bill Catchings, Bill Schilit, Frank da Cruz (Columbia University),
Davide Cervone (University of Rochester),
Paul Placeway (Ohio State University)
Matthias Aebi (ECOFIN Research and Consulting, Ltd.)
and many many others.

Language: MPW C
Documentation:
Version: 0.9(36)
Date: January, 1988

1. OVERVIEW

Macintosh Kermit, or "MacKermit", is an implemtation of the Kermit file transfer protocol for the Apple Macintosh computer, developed at Columbia University, based on C-Kermit (which also forms the nucleus of Unix Kermit).

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
Wildcard send:	Yes (folder send)
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
Transmit BREAK:	Yes
Support for dialout modems:	No
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:	Yes
File attributes packets:	No
Command macros:	No
Raw file transmit:	No

The main differences between MacKermit and other Kermit programs are:

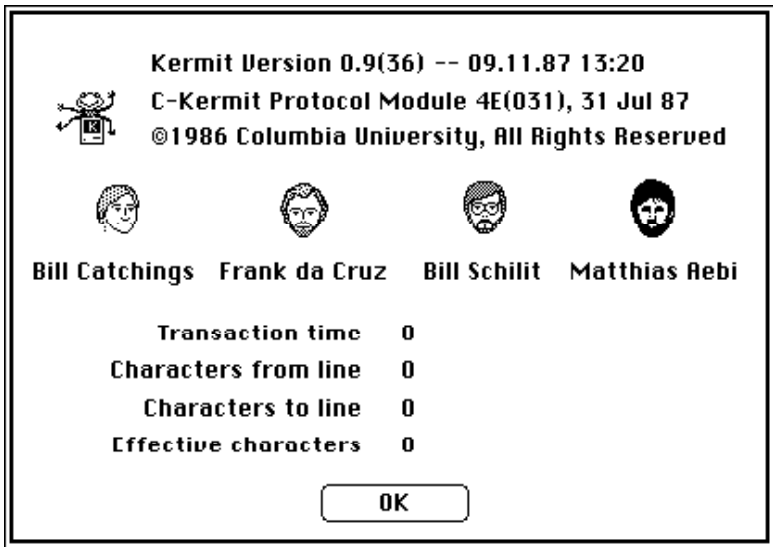
- In MacKermit you are always connected via a terminal emulator (VT102).

- MacKermit commands are issued by means of pull-down menus that overlay your terminal session.

2. MENUS

Besides the Apple Menu there are currently four main menus: File, Settings, Remote, and Transfer. The menus and all their items are discussed in the following sections.

About Kermit shows several informations about program version and some transaction statistics in the following dialog Box:



The top line displays the version number of MacKermit and the last compilation date and time.
The second line shows the version number of the protocol kernel which is being used in the program. The protocol kernel is a collection of machine independent C procedures. These procedures are responsible for all protocol related capabilities of MacKermit (like long packets or 8th-bit prefixing for example).

The four bottom lines show statistics about the last transaction (like sending or receving a file or a set of files). «Transaction time» is the time needed for the transaction measured in seconds. «Characters from line» counts all characters that the Macintosh receives from the serial line (including acknowledge packets for example). «Characters to line» is the same count for all characters sent through the serial port. «Effective charaters» counts the number of "useful" characters transmitted. This does not include any protocol overhead characters.

File

The File menu invokes MacKermit's file transfer functions, allows settings to be saved and loaded, and like most Macintosh applications, includes a «Quit» item for leaving the program.

Load Settings...

Displays a standard file dialog box which shows all settings files in the selected folder. Opening one of the files loads all important settings (like baud rate, parity, packet-length, and so on) including the definitions of special keys. See «Settings» menu description for more information. Loading a new settings file just overwrites the current settings. So in order to keep the settings you made, save them first with «Save Settings…». Please note that old settings files do **not** work with the new version of MacKermit. In the Finder you can distinguish them by the icon they use:


Old Kermit

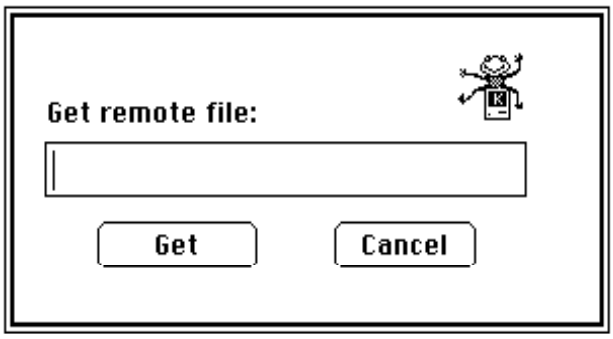

New Kermit

Save Settings...

Saves the current settings for communication, protocol, terminal, special keys amd so on in a file you can specify in a standard file dialog box.

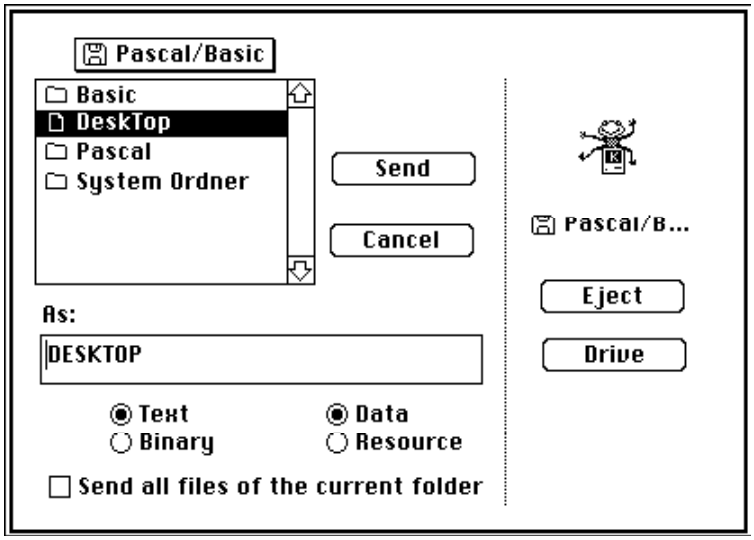
Get file from server... ⌘G

Sets up a dialog box where you can enter the name of the file to get from a remote server. If the remote server is a Macintosh too, you may enter a colon (:) to get all the files in the current folder.
Of course this also works if you run a Macintosh as a server to an IBM-PC for example.

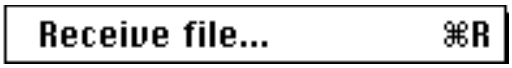


Send file... ⌘S

Sends a file (or all files in a folder) which can be chosen in the following dialog box:



You can either send one file at a time, by clicking on its name in the file list or all files in the current folder by checking the appropriate box. 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. If you type a name but send all files in the folder, the first file sent will be renamed to the chosen name. Multiple files are sent in alphabetical order. When you select a file, MacKermit examines its type; if the type is APPL, then MacKermit expects to send the resource fork in binary mod, 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.

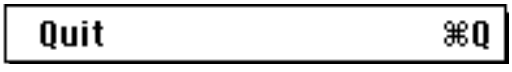


You can receive or get multiple files, providing the opposite Kermit is capable of sending multiple files in a single transaction (most are). As files arrive, they will be decoded according to the current mode (text or binary), and stored in the default fork (data or resource) under either the name they arrive with (overwriting existing files of the same names) or under 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 all file defaults on a per-file basis. But this option 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.

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 in the folder of the application. 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.

File transfers can be cancelled 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.

In any case, an "emergency exit" from any protocol operation can be taken at any time by typing "Command-." -- that is, hold down the Command (Fan, Cloverleaf) key and type period.



Quits MacKermit. Be careful if you would like to keep your current settings. MacKermit will not remind you to save them first. This is your responsibility. «Quit» also closes the session and transaction logging files if they are open.

Glossary:

- Mode - 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 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.

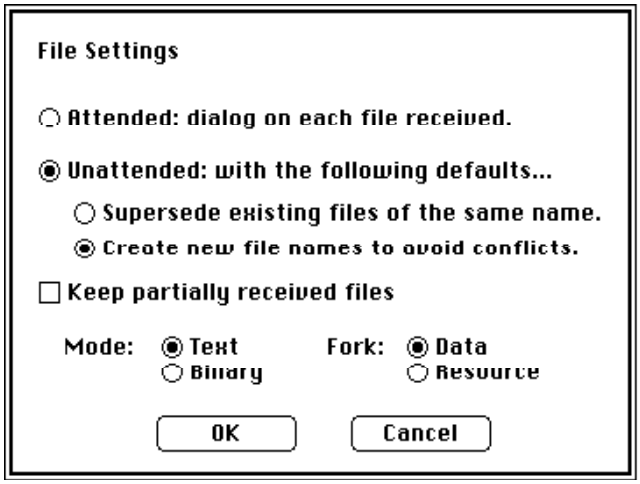
- Fork - Data or Resource. Macintosh files may have two "forks". The data fork contains data for an application; the resource fork contains icons, strings, dialog boxes, and so forth. For instance, a MacWrite document contains text and formatting information in the data fork, and fonts in the resource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

Settings

The Settings menu provides dialog boxes for file, communications, protocol and terminal settings. Besides calling up these dialogs it allows the redefinition of keys and key modifiers and switching on and off session and transaction logging. You can save and restore these settings by invoking the appropriate selection in the File menu. You can double-click on the resulting document to start MacKermit with those settings.

File Defaults...

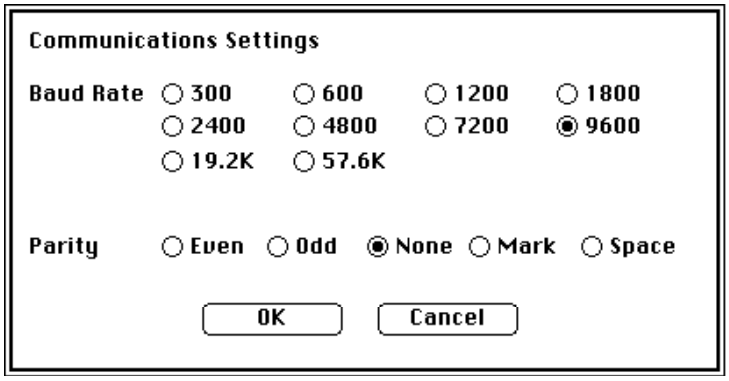
Puts up the following dialog box which lets you specify the default file parameters:



The File settings establish the defaults for file transfer:
- Mode: 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).
- Fork: which fork -- data or resource -- to send, or to store an incoming file into.
- Naming: 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. exists, then FOO.2, etc.
- Attended versus Unattended operation for incoming files.

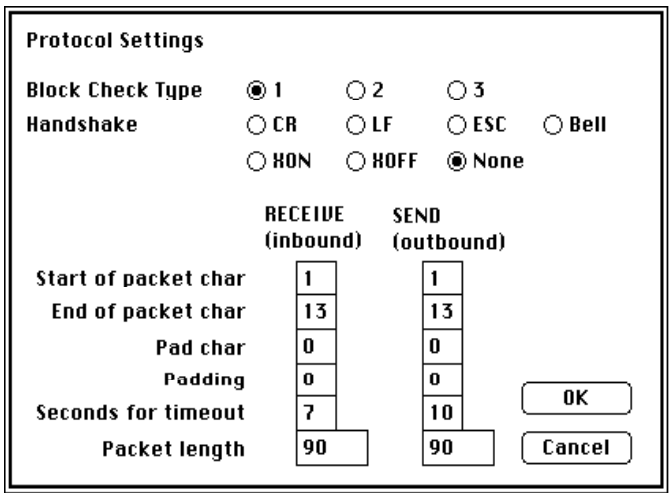
Communications...

The Communications settings dialog box allow you to set the baud rate (anywhere between 300 baud and 57.6K baud) and parity (odd, even, mark, space, or none).
***** explain mark and space parity here *****

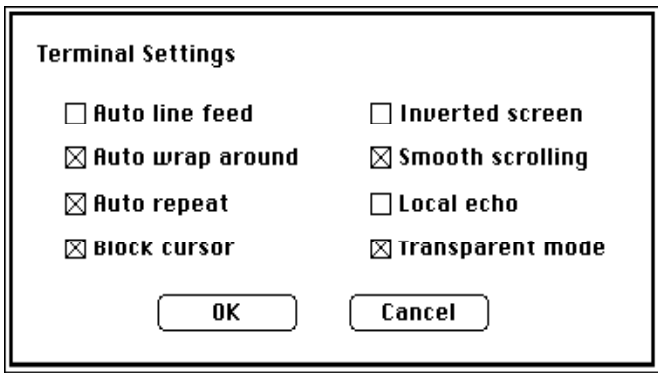


Protocol...

The Protocol settings allow you to set packet parameters for both incoming and outbound packets: These parameters 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 so forth (Refer to Kermit User Guide). Characters are specified by entering their ASCII value in decimal, e.g. 1 for Control-A, 13 for Control-M (Carriage Return), etc. MacKermit can send and receive long packets. You can make the opposite Kermit to send long packets (assumed it knows how to do it) just by setting the receive packet length to a number up to 1024. If long packets cannot be sent by the other Kermit, both sides will use standard packets.



Terminal...



***** use new screen snapshot *****
***** explain all checkboxes here *****

MacKermit provides a subset of the features of the DEC VT102 terminal; the VT102 is a VT100 with line and character insert/delete functions added. The functions provided are sufficient to allow MacKermit to act as a terminal for EMACS as it exists on the DEC-20, VAX (CCA EMACS on VMS

or UNIX), and for most host-resident display-oriented applications that expect to do cursor positioning and editing on the VT100 screen. MacKermit does not currently support the following VT100/102 functions (among others):

- double height or double width lines
- 132 columns
- Interpretation of multiple parameters in a single escape sequence ***** true anymore ? *****

MacKermit includes a mouse-controlled cursor postioning feature for use during terminal emulation. When the mouse button is pressed inside the terminal window, the program acts as if you typed the keypad arrow keys to move the terminal cursor to where the mouse cursor is.

MacKermit honors your parity communications setting by using built-in functions of the Mac's serial i/o chip. Unfortunately, the chip has an unpleasant quirk -- arriving characters that do not have the specified parity are discarded rather than passed to the requesting application. Thus, if you are connected as a terminal using MacKermit to a device that requires, say, odd parity on characters sent to it, but does not put odd parity on characters it sends to you, then many incoming characters will not appear on your screen. ***** true anymore ? *****

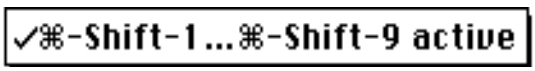
To allow useful coexistence of desk accessories and Kermit, the terminal emulation may be dragged using the drag bar. A desk accessory that overlays the Kermit window can be clicked upon to move it behind the Kermit window, and then the Kermit window can be dragged to reveal the hidden desk accessory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

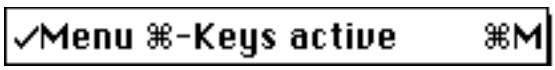
The following features are missing from the MacKermit terminal emulator, and may be added in subsequent releases:

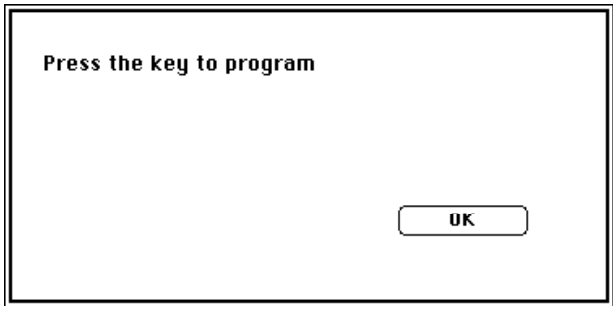
- capturing text from the screen (e.g. cutting to clipboard, saving off top)
- screen rollback, sizing
- modem or dialer control
- login scripts
- transmission of raw text to host (e.g. pasting to screen)
- printer support

MacKermit does not use XON/XOFF flow control during terminal emulation or file transfer. The terminal emulator can normally keep up at 9600 baud, but after several continuous scrolling screens at this speed, some characters may be lost. In the present version, when running at high baud rates keep your ter- minal in page mode, or use "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 will be lost and the display will become confused.

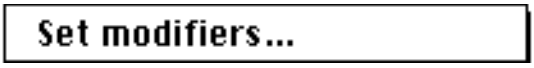
***** explain all following menu items in detail *****

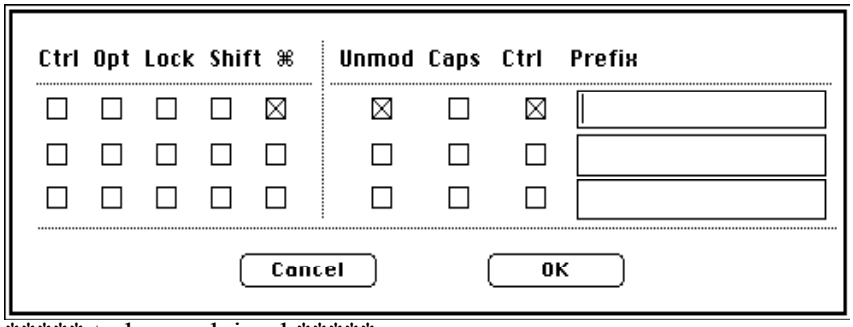




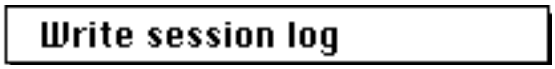


***** use new screen snapshot *****
***** to be explained *****

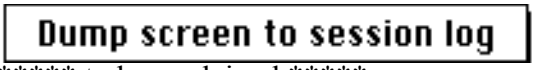




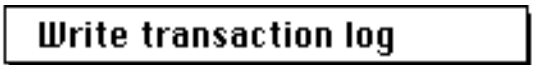
***** to be explained *****



***** to be explained *****



***** to be explained *****

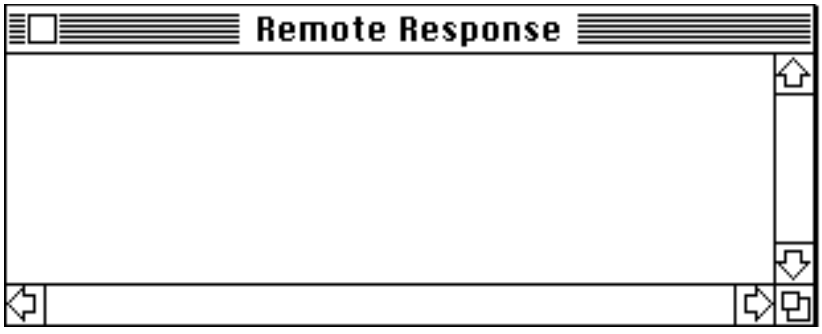


***** to be explained *****

Remote

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.

Show Response



The Remote menu allows you to send commands to a Kermit server. The response from these commands (if any) is displayed in a special 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 -- is up front.

If the response window gets too full (i.e. fills up the free memory available to the MacKermit application), the program will probably bomb.

If the remote Kermit server is in binary mode, its responses to Remote commands may look strange. For instance, a Unix Kermit server in binary mode will send lines of text separated by only linefeeds, rather than CRLFs.

A Remote command can be cancelled by taking the Emergency Exit (Command-.).

Finish
***** to be explained... *****

Bye

Cwd...

Delete file...

Directory...

Help...

Host...

Space...

Type...

Who...

Transfer

The Transfer menu gives you a standard Macintosh file box, allowing you to transfer directly to the selected application.

***** All following text is kept here to be integrated anywhere it makes sense *****

1.6. Installation
MacKermit is distributed in source form for building on Unix (or VMS/Eunice) systems that have the Stanford SUMACC Macintosh cross-development tools, in .HQX "binhex" 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 have the binary resource file available (its name will be CKMKER.RSRC, ckmker.rsrc, CKMKER.RSR, ckmker.rsr, 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, use MacPut to download the binary resource file to MacTerminal's XMODEM option on your Mac. After doing this you must use SetFile on the Mac to set the author to KERM, the type to APPL, and turn on the bundle bit. For CKMKEY, the author should be KERK.

If you have an earlier release of Columbia MacKermit, you may use Kermit in place of MacTerminal and MacPut.

If you don't have the binary resource file available, you can download the CKMKER.HQX file in the same manner, then run "binhex" (version 4) on it.

1.7. CKMKEY - Macintosh Kermit's Keyboard Configurator
CKMKEY is a keyboard configurator program for use with Macintosh Kermit (versions 0.8 and greater). CKMKEY allows:
- Redefinitions of keys
- Definitions of multicharacter function keys
- Selection of long and short BREAK keys

Some familiarity with the ASCII alphabet is assumed in the following discus- sion.

1.7.2. Modifier vs Normal Keys
The Macintosh keyboard is composed of normal keys and modifier keys. Modifier keys are SHIFT, CAPS LOCK, OPTION, and COMMAND (also known as APPLE, CLOVER, or FAN). Only one normal key can be selected at a time, but one or more modifier keys can be depressed along with it.

1.7.3. Key Maps
When a key on the keyboard or numeric keypad is depressed the result is a scan code -- a number between 0 and 127 (see Inside Mac Event Manager for details if you're interested). A table indexed by scan code resulting in the character to be displayed or transmitted will be referred to as a "keymap" or "key mapping."

On the standard Mac many keymaps exist -- the modifier keys (such as SHIFT) specify which keymap is selected. For example, when no modifer keys are depressed the keymap contains the lowercase alphabet, numbers and some punctua- tion. The keymap in use when the SHIFT modifer is depressed contains the capi- tal letters and special characters.

All in all it is possible to select 16 different keymaps by depressing from zero to four modifier keys. Normally however, 6 or so distinct keymaps will suffice.

CKMKEY allows you to redefine 6 keymaps: shifted and unshifted combinations of keymaps named "normal", "capslock", and "control". These keymaps are predefined with the expected values -- the control map is preloaded with con- trol codes, the capslock preloaded with the unmodifed keymap but with all letters uppercase.

In this document modifier keys are written in capital letters and key map names are written in lowercase. SHIFT, CAPS LOCK, COMMAND, and OPTION are modifier keys, "normal" "capslock" and "control" are key maps internal to CKMKER. Since one of the major functions of CKMKEY is to change maps invoked by modifier keys, it is important to keep this distinction in mind.

1.7.4. What's in CKMKEY's Keymaps
A keymap is a list of 128 numbers. Which keymap is selected depends upon which modifier keys are depressed, and the entry within the key map is determined by the scan code. A keymap entry is an 8-bit quantity: if the high order bit is 0, then the entry is the 7-bit ASCII character to be transmitted through the serial port; if the high bit is 1, then the remaining 7 bits are an index into the function-key table.

Notice that only single 7-bit values can be directly translated through the CKMKEY keymap. If you want a single key to transmit multiple characters, then you can designate that key to be a "function key", and the key map will contain an indirect reference to the function-key table. If you want a key to transmit an 8-bit value, assign the "meta" operation to one of the modifier keys and use the meta key together with the desired key (see below).

Functions are numbered 0-127 with the highest few being reserved for special use. Currently functions 126 and 127 send a short 250 millisecond BREAK signal and a long 3.5 second BREAK respectively. In the future more special functions may be allocated so (since it is arbitrary anyway) please use low numbered functions when defining your own.

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, and punctuation -- are the graphic characters. When a character is typed with the CONTROL modifier pressed, its "control equivalent" 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 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 ASCII value to be set to 1 upon transmission. META characters can only be transmitted when the communica- tion path allows all 8 bits to pass transparently; when this is not possible, software like 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" 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 have to use a "meta prefix" character (usually escape), and to enter META-F repeatedly in this case, you'd have to type <escape>F <escape>F <escape>F... etc.

Macintosh Kermit Modifier Keys:
You can define the modifier key to keymap correspondence in CKMKEY by selecting the "Modifer Keys..." menu item under SET, or by double clicking on a modifier key while in the SET KEYS dialog.

The SET MODIFIERS dialog lets you define what map OPTION, CAPS LOCK and COMMAND refer to. Notice that SHIFT is missing -- SHIFT always references the shifted equivalents to the normal, control and caps lock maps.

The dialog is layed out in columns with the three modifier keys as column head- ings and the three map names below. Also under each column is a "pseudo" key map for "meta."

Meta is not a map, but an operation: it augments the value being transmitted after it has been read from its map. Meta can either be set to send a prefix string before the character or to turn the high order (8th) bit on in the transmitted character. The default prefix for meta is set to be 033 (escape). If a meta modifier key is depressed and the key results in a function reference then no modification occurs; functions are not "metized". However, functions can be defined to include 8-bit values.
Notice that meta can be set in conjunction with a key map. Since meta is an operation as described above there is no ambiguity. Consider for example set- ting OPTION to reference the "control" map and selecting "meta" for this modifier key as well. The result is a control-meta key.

CAUTION: If you have used Kermit's communications settings menu to select any parity other than "none", then any high order bits that may be set by CKMKER's key mapping will be superseded when Kermit applies the selected parity to outbound characters.

The SET MODIFIER KEYS dialog also lets you select your meta preferences: whether you want to use the 8th bit toggled on, or a prefix string. The prefix string is entered in the same manner as a function definition (backslash fol- lowed by 3 octal digits for non-printable characters, see below).
Note that it is possible to cause ambiguities when selecting and using modifier keys. For example say you set OPTION to refer to the control map, and you set CAPS LOCK to refer to the caps map; at this point if you hold both OPTION and CAPS LOCK down it is unclear which map you want your character to come from. To try to prevent this type of ambiguity the SET KEY dialog will beep when you are holding down or mousing an ambigous set of modifier keys.

The Kermit code itself references maps in this precendence: if a control map modifier is depressed then use control map, else if a capslock modifier is depressed use capslock, otherwise use the normal map.

A sample modifier key configuration is shown in Figure -MACMKEY.

Figure 1-1: Macintosh Kermit Modifier Key Dialog

Here the CAPS LOCK key is used to reference "control", the COMMAND key to do the "meta" operation, and OPTION is "control-meta". Holding down COMMAND and CAPS LOCK together will also result in control-meta.

1.7.6.2. DIALOG: Set Function Definitions
Background:
Skip to next section if you know what function keys are.
Many popular terminals have "function keys" in addition to the alpabetic, numeric, punctuation, and modifier keys described above. Function keys are usually labeled F0, F1, F2, ..., or PF1, PF2, ... On some terminals, like the DEC VT100, the function keys send predefined sequences of characters -- for in- stance PF1 sends three characters: ESCAPE (ASCII 033), followed by "O", and "P". On others, the function keys may have arbitrary strings assigned to them. For instance, if you find yourself typing "Aaaarrrgggghhh!!! Sigh..." a lot, you can assign this string to function key F1, and then pressing the F1 key will cause this entire character string to be transmitted.

***** Build Indes completly new *****
Index
, Binhex 1
CKMKER 0
CKMKEY 1
Macintosh Kermit 0
META Key 2
Setfile 1

Table of Contents

1. MACINTOSH KERMIT 0
1.1. The Macintosh File System 0
1.2. File Transfer 0
1.3. Remote Commands 0
1.4. Settings 0
1.5. Terminal Emulation 1
1.6. Installation 1
1.7. CKMKEY - Macintosh Kermit's Keyboard Configurator 1
1.7.1. What is CKMKEY? 1
1.7.2. Modifier vs Normal Keys 1
1.7.3. Key Maps 1
1.7.4. What's in CKMKEY's Keymaps 1
1.7.5. Menus 2
1.7.6. MENU: Set 2
1.7.6.1. DIALOG: Set Modifer Keys 2
1.7.6.2. DIALOG: Set Function Definitions 2
1.7.6.3. DIALOG: Set Keys 3
1.7.7. MENU: File 3
1.7.8. CKMKEY Known Limitations, Restrictions, Bugs 3
1.7.9. Unlocking CAPS LOCK 3
Index i

List of Figures

Figure 1-1: Macintosh Kermit Modifier Key Dialog 2