

# Termulator™

Version 2.02

## Shareware Notice

Termulator is shareware. This means that you are welcome to try it out free for 30 days. If you like it and continue to use it, you must register by sending \$25 to:

Brad Quick  
P.O. Box 231  
Staatsburg, NY 12580

I thought seriously about sending Termulator out as crippleware, where you could use some features, but not all, but I know crippleware can be a drag, so instead I'm counting on your honesty to support Termulator. Many long hours have gone into this program already, and I have many new features planned for future versions. With your help, Termulator will fulfill its goal of being the telecommunications program of choice for the serious Macintosh user. Again, thanks!

Please distribute Termulator freely. Be sure to include this documentation file and sample setup.

\* \* \*

## The Termulator Philosophy

Termulator was designed to be powerful, adaptable, and easy to use.

A Macintosh program should be powerful enough to do what the user wants it to do. It should be adaptable to the environment in which a particular user operates. Above all, it should be eminently usable. Its use should be intuitive. These were my primary goals when creating Termulator.

\* \* \*

## Features of Termulator

- **User Programmable Macro Buttons** Easy to use macro buttons can do anything from typing text to opening a desk accessory and pasting to it. Macro commands allow the macro to do menu commands (like Hypercard doMenu), start and end capture, and start Xmodem transfers. The WaitFor command is particularly useful for logon sequences, etc. A macro can be set to run on startup. Macro buttons are resizable.

- **Split Window** This feature allows the user to view two portions of what has occurred in his terminal session. He can scroll back one window while the other continues to display incoming data. This is useful when the text is going by too fast to read. While it goes by quickly in one window, you can read slowly in the other. It is also useful for getting the name of a file to download after the list has already gone by; scroll back to the list in one window, copy the file name, and paste to the other window.
- **Copy and Paste to Modem**
- **VT100 Compatible**
- **TEXT, Xmodem, CRC, 1K blocks, MacBinary** file transfer protocols all supported.
- **ZMODEM** send, batch send, and receive all supported.
- **All Fonts Supported.**
- **Remembered Screens.**
- **Clock** with \$ amount.
- **User Configurable Keyboard** The user can define key combinations for arrows, backspace, etc. Any key can be mapped to produce any character, control key, or to start a macro.
- **Buffered Keyboard** This is useful when speaking interactively with other users.
- **Saves All Settings** All settings are saved to a user-specified setup file, including window size and location.

\* \* \*

## How to Use Termulator

### Macro Buttons

Macro buttons are located on the macro bar that divides the two windows. To edit a macro button, simply hold down the command key and click on the macro button you want to edit. A dialog box will come up that will allow you to name the macro button. Enter the text for the macro button, and select whether the macro will be run on startup (only one macro can be run on startup, but that macro can invoke other macros by using the DoMacro command as described below).

## Macro Triggers

Each macro has an optional trigger associated with it. Any time Termulator encounters a string that is identical to the trigger for a particular macro, that macro will begin executing. When that macro is finished, execution is resumed in the macro that was running (if one was running). For a trigger to work, it must be enabled. There is a check box available for enabling the trigger in the macro edit dialog box.

## Macro Text

The text for a macro will consist of plain text, control characters, and/or macro commands. Plain text is entered as plain text. It will be sent to the modem as it is typed. All carriage returns in the text are ignored, so you will have to use ^M if you want a carriage return sent (see next paragraph).

Control characters can be entered into the text by typing a ^ followed by a letter. For example, ^M would be used for a carriage return since control M is the equivalent of a carriage return.

To send a ^ or a \ character, precede it with a \. Type \\ to send a \, or \^ to send a ^.

To have the macro perform one of the special commands, enter \ followed by the name of the command. If the command requires a parameter, enter a space after the command then the parameter in either quotes or parenthesis.

The time and/or date can be entered automatically by using a \D or \T. These special commands can be entered in the plain text, or in a parameter string for another command. For example the command:

```
\CaptureFile "MyFile \D \T"
```

would start a file capture with the name "MyFile 1/6/90 9.14.55 PM" or whatever the current date and time is.

Commands can be entered into the text quickly by clicking on the command you want to enter while the cursor is in the text area of the macro editing screen. Commands may also be typed directly. They can occur anywhere in the text, and case is not important.

The commands that are currently supported are:

**\DoMenu** "parameter" where parameter is the name of a menu item. This executes a menu item just as if you had selected it in the menu bar. The menu item must be typed exactly as it appears in the menu bar (spaces and case are important). This can be used to open desk accessories, Termulator menu commands, and even macros from Apple's MacroMaker.

**\DoMacro** "parameter" where parameter is the name of another macro button. This starts the macro whose name is the parameter. When execution of that macro is completed, the calling macro continues where it left off. Macros can be nested up to 32 deep!

**\GotoMacro** "parameter" where parameter is the name of another macro button. This starts the macro whose name is the parameter. The difference between this and \DoMacro is that \GotoMacro stops all other macros before executing, so control never returns to the calling macro.

**\DoFile** "parameter" where parameter is the name of a file. This opens the file and treats it as if it were macro text. All commands are performed, control character interpreted, etc. If there is no parameter, a dialog box appears for user selection of a file.

**\SendFile** "parameter" where parameter is the name of a file. This starts a TEXT Send of the file. If there is no parameter, a dialog box appears for user selection of a file.

**\CaptureFile** "parameter" where parameter is the name of a file. This starts capturing text to the file. If there is no parameter, a dialog box appears for user selection of a file. If the file exists, the file is overwritten. If a file is already being captured, it is closed and the new file is opened.

**\CaptureAppend** "parameter" where parameter is the name of a file. This starts capturing text to the file. If the file exists, new text is appended onto the end of the file.

**\EndCapture** This ends a TEXT file capture.

**\SendX** "parameter" where parameter is the name of a file. This starts sending the file using Xmodem. If there is no parameter, a dialog box appears for user selection of a file.

**\RecvX** "parameter" where parameter is the name of a file. This starts receiving the file using Xmodem. If there is no parameter, a dialog box appears for user selection of a file.

**\SendZ** "parameter" where parameter is the name of a file. This starts sending the file using Zmodem. If there is no parameter, a dialog box appears for user selection of a file.

**\RecvZ** "parameter" where parameter is the name of a file. This starts receiving the file using Zmodem. If there is no parameter, Termulator determines the name of the file. This command is not needed if the Auto Zmodem Recieve is checked under the X & Zmodem Settings menu.

**\ResetClock** This resets the clock to zero. This is useful in a login macro.

**\StartClock** This starts the clock if it has been stopped with \StopClock.

**\StopClock** This stops the clock.

**\ClockRate (parameter)** Where parameter is the number of pennies per hour to set the clock to.

**\WaitFor** "parameter" where parameter is a string of text. This causes the macro to stop execution until the parameter string is received by Termulator. At that point, execution of the macro resumes. All Termulator operations work as normal while a macro is waiting for a string.

**\SetString "parameter"** This loads the user variable (called the string) with parameter. The following commands also operate on this variable.

**\AskUser "parameter"** This causes a dialog box to come up on the screen with an edit field available for the user to enter text in. The parameter is used as the prompt text in the dialog box.

**\TellUser "parameter"** This causes a dialog box to come up on the screen as a message for the user. The parameter is displayed in the dialog box.

**\ReadModem** This reads one line from the modem and puts it into the variable string. An optional parameter can be used that tells the command what characters should cause the command to stop reading. By default, it stops when either a Return or Linefeed are encountered. To have it stop reading at the next space or Return, put a space and ^M in quotes as the parameter: \ReadModem " ^M"

**\S** This substitutes the variable string similar to the way \T and \D work.

**\IfWait "parameter1","parameter2"** This works together with \Else and \EndIf to allow the macro to wait for either of the two parameters. If parameter1 is encountered first, the script directly after the \IfWait command is executed up until the \Else command, where it jumps to the \Endif command. If parameter2 is encountered first, execution jumps to the \Else command.

**\IfStringIs "parameter"** This works together with \Else and \EndIf to allow the macro to check the variable string. If the variable string is the same as the parameter, the script directly after the \IfStringIs command is executed up until the \Else command, where it jumps to the \Endif command. If the variable string is not the same as the parameter, execution jumps to the \Else command.

**\IfWaitTime "parameter",120** This command works the same as \IfWait, except that it waits for either parameter, or for 120 (or any number) of 60ths of a second to pass (120 is two seconds).

**\Goto "Label"** This command causes execution to jump to the location of \Label in the macro. Any name can be used as a label name as long as it isn't another command name. The label must be in the same macro (or file) as the \Goto for that label.

**\SetTrigger "parameter1","parameter2"** This sets the trigger for the macro whose name is parameter1 to the string in parameter2. The trigger for this macro is automatically enabled.

**\EnableTrigger "parameter"** This command enables the trigger of the macro name whose name is parameter.

**\DisableTrigger "parameter"** This command disables the trigger of the macro name whose name is parameter.

**\Line "1200,8,N,1,F"** This command changes the line settings for Termulator to those in the parameter string. The settings allowed are:

- 300 for 300 baud.
- 1200 for 1200 baud.
- 2400 for 2400 baud.
- 4800 for 4800 baud.
- 9600 for 9600 baud.
- 19200 for 19200 baud.
- 8 for 8 data bits.
- 7 for 7 data bits.
- N for no parity.
- E for even parity.
- O for odd parity.
- 1 for one stop bit.
- 2 for two stop bits.
- F for full duplex.
- H for half duplex.
- > for echo incoming data back to modem.
- < for don't echo incoming data back to modem.

Any number of settings are allowed. For example, \Line "2400" will change the baud rate to 2400 and leave everything else as it was.

**\Echo "parameter"** where parameter is a string of text. This causes the parameter string to be displayed on the screen as if it were incoming data. The string is NOT sent out the serial port. This is useful in a macro to inform the user of what is going on in the macro.

**\Delay "parameter"** where parameter is a number. This causes the macro to stop execution for parameter sixtieths of a second (30 would be a half a second).

**\Beep** This makes your computer beep.

**\Play "parameter"** where parameter is the name of a "snd" resource installed in Termulator or in the System File. This plays the sound named in the parameter. You can add sounds to Termulator using a resource mover such as ResCopy or ResEdit. There are programs available to convert other types of sounds to "snd" resources so they can be used in Termulator.

**\StringToGlobal "parameter"** where parameter is the name of a global variable. This takes the value of the \S variable string and copies it into the global variable named in the parameter. If the global variable doesn't exist, it is created. Up to 32 globals can be used in any one session with Termulator. See XCMDS and XFCNS and Global Variables below.

**\GlobalToString "parameter"** where parameter is the name of a global variable. This takes the value of the global variable named in the parameter and copies it into the \S variable string. See XCMDS and XFCNS and Global Variables below.

In any of the macros that use a filename for the parameter, pathnames can be used to determine which folder the file resides in. If a default folder is set, Termulator will look there for the file, otherwise, the folder where the current setup is stored is used as the default.

To stop a macro in progress, choose the menu selection Cancel Send. Clicking on another macro button will also stop any macro in progress. This may be necessary if you have a macro call itself either directly or indirectly.

Once macros have been written, you can move them around by command-clicking on them and dragging them into another macro location. The two macros are swapped so you don't lose the old macro.

## **Split Screen**

To split the screen, click on the black bar on the right hand side of the macro buttons bar and drag the macro bar up or down.

Each window has its own set of scroll bars. These scroll bars control how far the window is scrolled back. You can scroll back one or both windows. If both windows are scrolled back and data is received by Termulator, whichever window is bigger will automatically be scrolled back to the bottom so that the incoming data can be seen.

If one window is scrolled back, and the other is receiving data, the thumb of the scroll bar that is scrolled back will slowly move toward the top of the scroll bar. When the thumb reaches the top of the scroll bar, and the data that is being viewed in that window is about to be lost, the window will be automatically scrolled back to the bottom. The more save screens that are set, the slower the thumb will move toward the top of the scroll bar.

Text selection can be done in either window. Automatic scrolling and shift clicking are supported. You can even start selecting in one window, then shift click to finish selecting in the other window.

The macro bar may be placed at the extreme top, or at the extreme bottom of the screen, so that only one window is visible.

## **Line Settings**

Typical line settings can be selected. In this version, only the modem port is supported.

## **Keyboard Settings**

Any key can be mapped to any character. In the Keyboard Settings dialog box, type the key you want to set (with command, shift, option, or control keys held down, if desired). The key code for that key combination will appear in the upper left of the dialog box. Next, click on the appropriate radio button to select what you want the key that you typed earlier to do. You can select either a macro or an Ascii value.

Be careful with this since it will allow you to change the function of an ordinary key (you can make the J key produce a K).

The Control and Option keys perform the same function, but on some Macs, some of the option combinations are intercepted by the system, so on Macs with a control key, it's best to use the control key.

Some keys are initially set for you. The Enter key is set to an ESC key. The control characters can be used by using the control or option key. Of course, these can all be changed.



Keys can be set to start macros, which may be helpful in allowing you to make use of some of the extra macro buttons that may not normally appear on your screen.

## **Terminal Settings**

Various terminal settings can be selected here. The number of saved screens can be changed. Normally, the more the better, but more saved screens require more memory.

## **Printer**

Echo to printer can be turned on and off under the Terminal Settings menu. Termulator will also recognize the VT100 escape sequences to turn the printer on and off automatically. You can also choose Print Screen Data from the Other menu. This causes the 80x24 screen to be printed.

## **Clock Settings**

You can turn the clock on or off, and you can set the rate (in pennies per hour) if you are using a pay per hour service.

To reset the clock, click on the clock. The clock can also be reset from a macro by using the \ResetClock macro command.

## **Default Folder**

Setting the default folder causes Termulator to look in that folder whenever a file is referenced via a macro command.

## **Color**

If you have a color monitor, you can set some limited colors on Termulator. If you have black and white, all non-white colors come out black, so the only real way to customize the color is to change the white to black and black to white.

## **Fonts**

Termulator will allow you to choose any font that is installed in your system file as the display font. Proportional fonts will not work very well when any special screen formatting is used (like vt100), so you may wish to use a monospaced font to begin with, like Monaco or Courier.

You can also choose the font size. Keep in mind that a normal VT100 terminal has an 80 x 24 screen, so if you choose too large a font, some of the characters may not be displayed in Termulator's window.

You can compress the font by choosing the Compress Font menu item. Compressing a font makes any font into a monospaced font by starting with the width of the widest character in the font and compressing it by the percentage entered by the user. All compression is done by chopping off the right hand side of each character.

Kerned (overlapping) fonts don't work correctly with Termulator.

On a Mac II with the Apple 13" color monitor, my brother likes to use the Courier font, 14 point, with zero compression.

## **Setup Files**

Most of Termulators settings can be stored in Setup Files and recalled at a later time. "Save Current Setup" and "Load Saved Setup" save and load these settings to and from files. Double Clicking on on of these Setup Files in the Finder will launch Termulator with the settings stored in that file. When Termulator is launched without a settings file, it looks for the Default Setup File named "Termulator Setup".

## **XCMDs and XFCNS**

Termulator supports the use of HyperCard external commands and functions. To use these, they must first be installed into Termulator using a resource mover such as ResCopy or ResEdit just as is done with HyperCard. To access the external command or external function, just type a backslash followed by the name of the command or function, followed by any command or function parameters separated by commas. For example, to call the Talk command, enter the following in a macro:

```
\Talk "This is a test",150,110
```

An external function is called the same way, but the result of the function is placed in the \S string. The following echos to the screen the result of a Pop Up Menu external function:

```
\PopUpMenu "item1,item2,item3",0,50,50  
\Echo "The result of the pop up menu is \S"
```

Of course, not all HyperCard externals will work with Termulator. Any command that tries to access fields or cards in HyperCard is likely to fail. If you experiment with external commands and functions, be prepared for a bomb or two as you figure out which ones work and which ones don't.

If the external command or external function uses HyperCard Global Variables, these globals can be accessed with the following commands.

## **Global Variables**

Your macros can store and read Global Variables that are created by your macros, or by XCMDs or XFCNS. To set a global variable to a particular value, use the \StringToGlobal command. You can retrieve the global with the \GlobalToString command. These commands are explained above. You can also use a global in a macro by typing a backslash followed immediately by the global name in quotes. For example to echo the contents of the global "MyGlobal" to the screen, type the following in a macro:

```
\Echo "\"MyGlobal\""
```

## **Memory**

Macro scripts are stored in memory while Termulator is running. Global variables are also stored in memory. If your setup has a lot of long macro scripts or uses a lot of Globals, you may need to increase the memory allocated to Termulator by MultiFinder.

\* \* \*

Thanks to all of the Termulator users who have made good suggestions that have been added to Termulator.

Special thanks to Andy Burns for the new Termulator Icons.

Please continue to send me ideas and/or complaints. You can send U.S. mail to:

Brad Quick  
P.O. Box 231  
Staatsburg, NY 12580

also,

Termulator is supported on Genie on the Mac RT, Category 5, Topic 63.

2/14/90 BSQ