TermTalk Script Language Reference

for

Minsoft for the Mac, Minisoft Pocket 92, and Minisoft 92 Secure

version 5.0

Minisoft, Inc.
1024 First Avenue
Snohomish, WA 98290
U.S.A.

1-800-682-0200
360-568-6602
Fax: 360-568-2923

Minisoft Marketing AG
Papiermühleweg 1
Postfach 107
Ch-6048 Horw
Switzerland

Phone: +41-41-340 23 20
Email: info@minisoft.ch
http://www.minisoft.ch

Internet access:
sales@minisoft.com
support@minisoft.com
http://www.minisoft.com

Disclaimer

The information contained in this document is subject to change without notice.
Minisoft, Inc. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.
Minisoft, Inc. or its agents shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishings, performance, or use of this material.

First printing September, 1997.

License Agreement

In return for payment of a one-time fee for this software product, the Customer receives from Minisoft, Inc. a license to use the product subject to the following terms and conditions:

- The product may be used on one computer system at a time: i.e., its use is not limited to a particular machine or user but to one machine at a time.
- The software may be copied for archive purposes, program error verification, or to replace defective media. All copies must bear copyright notices contained in the original copy.
- The software may not be installed on a network server for access by more than one personal computer without written permission from Minisoft, Inc.

Purchase of this license does not transfer any right, title, or interest in the software product to the Customer except as specifically set forth in the License Agreement, and Customer is on notice that the software product is protected under the copyright laws.

90-Day Limited Warranty

Minisoft, Inc. warrants that this product will execute its programming instructions when properly installed on a properly configured personal computer for which it is intended. Minisoft, Inc. does not warrant that the operation of the software will be uninterrupted or error free. In the event that this software product fails to execute its programming instructions, Customer's exclusive remedy shall be to return the diskette to Minisoft, Inc. to obtain replacement. Should Minisoft, Inc. be unable to replace the diskette within a reasonable amount of time, Customer shall be entitled to a refund of the purchase price upon the return of the product and all copies. Minisoft, Inc. warrants the diskette upon which this product is recorded to be free from defects in materials and workmanship under normal use for a period of 90 days from the date of purchase. During the warranty period Minisoft, Inc. will replace diskettes which prove to be defective. Customer's exclusive remedy for any diskette which proves to be defective shall be to return the diskette to Minisoft, Inc. for replacement.

ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS IS LIMITED TO THE 90-DAY DURATION OF THIS WRITTEN WARRANTY. Some states or provinces do not allow limitations on how long an implied warranty lasts, so the above limitation or exclusion may not apply to you. This warranty gives you specific rights, and you may also have other rights which vary from state to state or province to province.

LIMITATION OF WARRANTY: Minisoft, Inc. makes no other warranty expressed or implied with respect to this product. Minisoft, Inc. specifically disclaims the implied warranty of merchantability and fitness for a particular purpose.

EXCLUSIVE REMEDIES: The remedies herein are Customer's sole and exclusive remedies. In no event shall Minisoft, Inc. be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Table of Contents

Chapter 1
Getting Started

Introduction

This chapter describes the basic capabilities of a TermTalk script, and tells you how to use the commands on MS92's Script and the Settings menus to create and run scripts.

What scripts can do

A script is a sequence of statements written in TermTalk. When executed, a script automatically performs functions that would normally be performed by a user. For example, a script can:

- Dial Modems.
- Log on users.
- Set configuration values.
- Run applications.
- Transmit and log data.
- Create and delete files (on personal computer or host).
- Print files.
- Send and receive files to and from other computers.

- Prompt the user for input and receive input; display messages.
- Take different actions in different situations.
- Create and manage local screens; provide a user interface for host or PC-based applications.

Though TermTalk is capable of performing very sophisticated functions, new users can easily set up simple scripts to automate routine tasks.

How scripts are created

A TermTalk script can be created in one of three ways. You can:

- Turn on MS92's command recording facility while you perform a series of actions—the commands required to reproduce these actions are recorded automatically.
- Write and edit scripts in a MS92's script window.
- Use any text file containing valid TermTalk commands.

You can turn on MS92's automatic command recorder by choosing **Record a Script** from the Script menu. You then perform the function you want to automate while MS92 "watches" you and records the commands required to reproduce these functions to a script file. Or, if you would like to see the TermTalk commands displayed on the screen as you record them, you can choose **Show Script Window** to open the script window, and then turn on recording by choosing **Record a Script** from the Control menu of the script window. Scripts created automatically can be edited if necessary, then saved and re-used time and time again. More details follow in "Script Recording," page 4.

To write a script, choose **Show Script Window** from the Script menu. This opens the script window, which has its own menus with commands used to create and edit script files.

[p3]

Also, any standard text file containing TermTalk commands can be run as a script.

Executing scripts

TermTalk scripts can be executed in several ways. You can:

- Choose **Do Script** from the MS92's Script menu, and then choose the desired script in a dialog box.

- Place the names of the frequently used scripts on the script menu, so you can choose them directly.
- Associate the names of frequently used scripts with the function keys, so they are executed when you press or click that key.
- Designate scripts to be performed automatically when you open and close your settings file.
- Execute one command at a time by choosing Do **Command** from the Script menu.
- Execute the current contents of the script window, or any portion of it, using commands from the Control menu.
- Write a script that executes another script.

To perform a script that has been previously written and saved in a file, you choose **Do Script** from the Script menu. This opens a dialog box from which you can choose a script to execute. While the script is running, you can use the **Stop Script** and **Pause Script** commands to interrupt the script.

You can place the names of frequently used scripts on the Script menu so they can be run with one click of the mouse. Each settings file can have a different list of scripts on the menu (settings files are created by choosing **Save** or **Save As** from the file menu). To select the scripts to be displayed on the menu, choose **Scripts** from the Settings menu. The dialog box shown below is displayed. In this dialog box you can also specify a script to be executed automatically as the file is opened, and another to be executed as it is closed.

[p4]

To associate a script with a function key, choose the **Function Keys** command from the Settings menu. You can associate a script with a particular function key by selecting **Script** in the function column, then clicking the **Script** button to select a particular script file. Like other user-created function key definitions, function key scripts are removed when a host application loads the function key values and labels.

[p (1) 5]

When you choose **Do Command** from the script menu, a dialog box is displayed in which you can enter and execute one command at a time. This facility can be used to perform simple actions or to test commands which are to be placed in a script.

[p (2) 5]

When a script is open in the script window, you can execute single lines, selected portions, or the whole of that script using the Control menu commands **Do Line/Do Selection** and **Do Script Window.**
A script can also be executed by another script, by placing the do script command in a script file. Script can be nested in this fashion up to fifteen levels deep.
Each host session you run can execute only one script at a time. If you are running multiple sessions, they can all execute scripts simultaneously. When a script is operating within a particular session (called focusing on that session), the user is locked out of the session until the script terminated or changes its focus to another session.
There is an indicator in the lower left hand corner of the main Session window which shows you when a script is being executed. This indicator has four states, as shown below.
[p 6]

Compiling scripts

However, to streamline memory usage and to eliminate the possibility of an unforeseen compile time error, you may wish to compile scripts which will be called from other scripts before they are actually used in a production environment. To do so, use the TermTalk compile command. Syntax for this command is on page 67. Note that simply saving a script with the **Save** or **a Save As command from the File menu of the script window does** not compile it.
Like source programs written in general programming languages, TermTalk scripts must be compiled into object format before they are executed or used by the Host Control Facility (see page 201). Normally, you don't have to concern yourself with this compilation as it is performed automatically the first time a script is run.

Script recording

You can turn on MS92's automatic recorder by choosing **Record a Script** from the Script menu. (This command changes to **Stop Recording** after it is selected). Once recording is on, you perform the function you want to automate while MS92 "watches" you. The TermTalk statements needed to reproduce these functions are recorded in a file.
When you choose Stop Recording, a dialog box is displayed in which you can name and save your script file.

To suspend recording temporarily while you are creating a script (for example, if you need to access another PC application), choose **Pause Recording** from the Script menu. This menu command then changes to Resume Recording. When recording is paused, the indicator in the lower left of the icon indicates "Paused". When you resume recording, commands are appended to the end of the file, where you left off.
To view and edit a script file created in this way, choose **Show Script Window** from the Script menu and choose **Open** from the script window's File menu to display the script.

Recording into the script window

To record a script directly into the script window, so that commands are displayed on the screen as they are recorded, choose **Show Script Window** from the Script menu, then start recording by choosing **Record a Script** from the Control menu of the script window. Do not select **Record a Script** from the Script menu of the main window, as this records commands to a file instead of into the script window.
When the script window is open and recording is in progress, **Record a Script** changes to **Stop Recording**. You can suspend or stop recording at any time by choosing this command. You can then save the file, discard the existing data, reposition the cursor, open a new file, and so on. If you choose **Record a Script** again, recording resumes at the current cursor position. You can edit the script in the window at any time, and save it using the **Save** or **Save As** commands from the File menu. If you attempt to close the window without first saving the script, a dialog box is displayed in which you can name and save the file or discard it.

Recording steps

To create a script that logs on to a host computer and then performs a **Home Up**, **Clear Screen**, you would follow these steps:

1. Choose **Record A Script** from the Script menu.
2. Choose **Terminal** and / or **Connection** from the Settings menu to configure any communications parameters you need. Then type your logon and click the **Home Up** and **Clear Display** buttons.
3. Choose **Stop Recording** from the Script menu. Save your script, specifying a name in the dialog box displayed.

4. Log off, and then test your script by choosing **Do Script** from the Script menu. If it does not work as you expected, you can either open the script window to take a look at the script, or you can simply repeat the whole process in case you made a mistake the first time.
5. If you wish to have this script executed automatically when you open your settings file, you would choose the **Script** command from the Settings menu of the main MS92 window. You could also choose to have your logon script placed on the script menu by making the appropriate settings in this box.

To record the same script with the script window open, you might follow these steps.

1. Choose **Show Script Window** from the Script menu, and then choose **Record a Script**. Then click in the main MS92 window so that it becomes the active window again. You may wish to size the two windows so they are both visible on you screen.
2. Do the operation you want to record as described above. Commands appear in the script window as you perform each action.
3. Test the script by choosing **Do Script Window** from the Control menu. Edit the script as desired.
4. Choose **Save As** from the File menu, and type a file name for your script.

There are some things to look out for when using automatic command recording. For example, if you iconize MS92's window, the window becomes an icon, and you must reopen it to continue creating your script. This activity gets recorded too, whether you like it or not. You can edit the script later to remove any unneeded commands.

Also, if you record a script which performs complex interactions with host software, this script will very likely require editing to fine-tune timing and handshaking procedures. You will probably need to use the TermTalk expect command to condition the sending of data upon the reception of the appropriate prompt from the host.

Using the Script Window

MS92's script window is a built-in text editor with which you can open, create, edit, and save scripts. To open the window, choose Show Script Window from MS92's Script menu. To close the window, choose Hide Script Window.
The title bar of the script window says Script Editor (or the name of the script file being edited).
[p 9]
The lower left corner of the window contains three indicators. The first shows the current activity. "Compiling" means a script is being translated into object format. "Running" means a script is being executed. "Paused" means that script execution has been suspended.
"Recording" means that command recording is in progress. "Editing" means that you are neither recording nor executing a script, and can add to or edit the contents of the window as you like.
The second indicator displays the current line number.
The third indicator displays the title of the focused session (the session from which you opened this window). The title is configured in the Terminal Settings dialog box (choose Terminal from the Settings menu of the main MS92 window.)
The window has four menus: File, Edit, Font, and Control. The commands on these menus are described below.

File menu

The File menu contains commands used to manipulate script files.
NEW [???]
Closes current file; creates and opens a new one.
OPEN
Presents a dialog from which you can select a MS92 script file or text file to be opened. The current script file is closed and the one you select id opened.
SAVE
Saves the contents of the current window to the file named in the title bar. If the window still has the default name, a dialog box is presented in which you can enter a different file name and select the drive on which the file is to be located.
SAVE AS

Saves the contents of the current window, presenting a dialog box in which you can enter a file name and select a disk drive.
SAVE AS ENCRYPTED
[???]
REVERT
Returns all settings to the values last saved in the current settings file, or to MS92's defaults, or to MS92's defaults if no settings file is in use.
PAGE SETUP [??? pg.11]
Presents a dialog box in which you can specify paper size, orientation, printing effects, and so on.

Edit menu

The Edit menu contains commands used to manipulate text in the script window.
UNDO
Undoes the effect of the immediately previous insertion or deletion of text.

CUT
Removes selected text from the window and places it on the Clipboard.
COPY
Copies selected text, leaving it in the window, and places it on the Clipboard.
PASTE
Places the text on the Clipboard in the window, starting at the current cursor location.
CLEAR
Removes the selected text from the window.
SELECT ALL
Selects all the text in the window.
PASTE COMMAND
Presents a dialog box in which you can choose a command from a list of TermTalk commands. The selected command is pasted into the script window with its default syntax, where it can be modified as desired.
GO TO LINE

Presents a dialog box in which you can enter a line number. When you click OK, the window is scrolled so the selected line is at the top of the screen, with the cursor positioned before the first character.

FIND

Presents a dialog box in which you can enter a string. When you click **Find Next** or **Find Prev**, MS92 searches through the script window for a matching string, beginning at the point where the cursor is located. **Find Next** searches forward through the window; **Find Prev**, searches backward.

Normally, a matching string need not be in the same combination of upper and lower case letters. To restrict the search to a string of identical case, choose the "Case Sensitive" option.

FIND NEXT

Searches forward through the script window for the next occurrence of the last string specified in the Find dialog.

FIND PREV

Searches backward through the script window for the previous occurrence of the last string specified in the Find dialog.

Font menu

Under MS92 for Mac, you can choose any of the fonts in your system suitcase. The character's size and style are selected under those menus.

HP SCREEN

The screen font of the MS92 application. Includes Roman-8 international characters.

ANSI

The standard ANSI character set.

7, 12, 16

The default font size is 12.

Control menu

The Control menu contains commands useful in creating and debugging script.

DO SCRIPT WINDOW / STOP SCRIPT

Executes the entire contents of the script window. While the script is executing, this command reads **Stop Script**, and all other commands are disabled except **Pause Script**.
DO LINE OR DO SELECTION / STOP SCRIPT
Executes the line in which the cursor is currently placed. If text has been selected, this command reads **Do Selection**, and choosing it executes the highlighted portion of the window. While the line or selection is executing, this command reads **Stop Script**, and all other commands are disabled except **Pause Script**.
PAUSE SCRIPT / RESUME SCRIPT
Pauses execution of the script initiated with **Do Script Window, Do Line,** or **Do Selection**. Once you have selected **Pause Script**, the command changes to **Resume Script**. Choose **Resume Script** to continue execution.
RECORD A SCRIPT / STOP RECORDING
Monitors actions performed by the user and places TermTalk commands which reproduce those actions in the script window. Once recording is in progress, this command changes to **Stop Recording**.
CHECH SYNTAX
Checks the syntax of the commands in the script window. If a portion of the text has been selected, the highlighted text will be checked, rather that the entire window. If an error is found, MS92 displays a dialog box showing the line number and the error message. This command finds only one error at a time, so after correcting the first error located, you should choose **Check Syntax** again.

Help menu

The Help menu accesses MS92's online help faculty.
[p.14]
Choosing any command on the Help menu opens a window in which you can get online information about MS92. The Help window has its own menus and buttons. For a complete description, choose any Help menu option to bring up the window, and choose **Using Help** for a complete explanation of general help procedures.
[p. 15]
Choose **Help Index** on MS92's help menu (or click the **Index** button) for a complete list of topics, or choose any one of the specific topic areas listed to go directly to that information.

Style and Size menu

In MS92 for Mac, the Style and Size menus allow you to select character style and size.

Chapter 2
Building TermTalk Scripts

Introduction

This section describes the basic elements of the TermTalk script language, and explains the rules which govern their use. To illustrate most of the elements of the language, statements from the following sample script are used.

```
;This script purges HP 3000 files specified by the user.
Input into filespec prompt&
  "Enter fileset to purge, using @ and ?:"
sendline "LISTF " + filespec + ",2"
expect "^Q" into list_of_files
if list_of_files CONTAINS "NO FILES FOUND"
  message prompt "No files found." Icon warn buttons "OK"
  return;
endif
;  Start reading at line 7 to skip the "listf" command that is
;  echoed back fromthehoast and the6 lines of header info printed
;  by LISTF
I = 7
while I <=numberOfLines (list_of_files) and line I of&
                list_of_files <> ""
```

```
    current_file = word 1 of line I of list_of_files
    sendline "PURGE" + current_file
    I = I + 1
  endwhile
```

This script prompts a user to specify a fileset to be purged on the HP 3000. After the user's input is received, a LISTF command containing the file specification is sent to the HP 3000. If the files specified do not exist, and error message is displayed in a dialog box. If the files do exist, the names of the individual files are extracted from the listing produces by the LISTF command, and these files are purged one by one.

Individual lines and sections of the preceding script are reproduced in the pages that follow to illustrate various aspects of MS92's script language.

Commands

At the heart of TermTalk are approximately 85 commands, which perform the following kinds of tasks.

Operation	Examples
User Interaction	beep, input, message
Configuration	set, get, configure, hide
Host Interaction	break, delete char
File Transfer	send, receive
Working with Files	open file, read, save
Controlling MS92	open application, quit
Editing Data on Screen	clear, cut, copy, paste

Each TermTalk command is described in the alphabetical reference in Chapter 4.

Commands can be continued over multiple lines by placing the ampersand character (&) at the end of each continued line. Ampersands can not be placed in the middle of a quoted string, however.

In the sample script shown at the beginning of this chapter, four commands are used: input, sendline, expect, and message.

Input into filespec prompt "Enter fileset to purge, using @ and ? :"

The input command displays a dialog box in which the user is prompted to enter a fileset specification:

[p. 18 (1)]

The user's input is stored in the variable filespec.

sendline "LISTF " + filespec + ",2"
expect "^Q" into list_of_files

This sendline command incorporates the user's input into a LIST,2 command which is sent to the host. The expect command places the host's output into the variable list_of_files until the specified terminator, ^Q, is received. ^Q (DC!) is the HP 3000 prompt character.

message prompt "No files found. " icon warn buttons "OK"

The message command displays a dialog box containing a user message. A warning icon is used to identify the message as an error condition, and an **OK** button is provided.

[p. 19]

A second sendline command builds an MPE PURGE command each specified file, and sends it to the host.

sendline "PURGE " + current_file

Variables

Variables play the same role in TermTalk script as in other programming languages – they are placeholders set up to contain information, either strings of text or numbers. You establish variable names as shown in the input command from the sample script at the beginning of this chapter.

```
Input into filespec prompt "Enter fileset to purge, &
  using @ and ?:"
```

filespec is a user-defined variable into which the file specifications entered by the user will be placed. In the commands which follow input in the example,

```
sendline "LISTF " + filespec + ",2"
expect "^Q" into list_of_files
```

the contents of file spec are incorporated into the LISTF command string sent to the HP host. list_of_files is a variable as well, one which will receive the output of the file directory listing returned by the HP host. (Note that the +sign is used to assemble portions of a text string. This is called concatenation, and is discussed in the section titled "Strings" on page 25.)
In the preceding examples, variables are specified as part of a command. The following script also includes variables which are set up and used independently, rather than as part of a command:

```
I=7
while I <=numberOfLines (list_of_files) and line I of
                list_of_files <> ""
  current_file = word 1 of line I of list_of_files
  sendline "PURGE" + current_file
  I = I + 1
endwhile
```

Here, the variable I is used to track progress through the files listed in list_of_files. Also, the variable current_file is used to store the name of the file being purged. Whenever numeric variables are used, the **must** be initialized, as in the example above.

Variable names

Variable names can include upper and lower case letters as well as numbers and the underscore character, and may be up to 31 characters in length. The first character must be a letter or an underscore. A total of up to 255 characters may be used without causing an error, but only the first 31 compose the unique name which identifies the variable.
If the variable has the same name as a TermTalk command, function, configuration setting, or other system-defined term, it must be terminated with a # symbol to distinguish it from that term. For example, if you wish o use input as a variable, you must specify input#, because input is a command. If you like, you can terminate all you variables with #. This also prevents any possibility of conflict with TermTalk's many system-defined terms. For a complete list of system-defined terms, see Appendix A, page 183.
In addition to avoiding conflicts with system-defined terms, you must also avoid giving a variable the same name as other user-defined terms. These include user defined procedure names established with the proc statement, and go-to section names established with the label statement. See "User Defined Procedures," page 36, and "Logic and Flow Control," page 32.

Permanent variables

Normally, the value of a particular variable is stored until the end of the script in which it occurs. If a variable with the same name is used in a different script, it will not pick up any value assigned to a variable with the same name in a previous script. You can, however, establish a permanent variable which remains defined throughout your entire session, and can be accessed by a number of different scripts. To do so, use the permanent statement, as shown below:

```
permanent variable_name
```

Declarations of permanent variables must be first statements in a s script, preceded only by comments. The following two scripts illustrate the use of permanent variables.

```
;Script 1
permanent variable1, variable2
;Declare permanent variables
variable1 = “New York”
variable2 = “Los Angeles”
display “The values at the beginning of Script 1 are :^M^J”
; Show the values of the variables
display variable1 + CR + LF
display variable2 + CR + LF + CR + LF
; Execute Script2 and continue
do script “Script2”
display “The values at the end of Script1 are: ^M^J”
; Show the values of the variables again
display variable1 + CR + LF ;
display variable2 + CR + LF ;
;Script 2
;  Declare variable1 permanent
permanent variable1
variable1 = “Berlin”
variable2 = “Munich”
display “The values in Script 2 are: ^M^J”
display variable1 + CR + LF
;  Show the values of the variables
display variable2 + CR + LF + CR + LF  ;
```

If you were to run Script 1, the output would look like this:

```
The values at the beginning of Script 1 are:
New York
Los Angeles
The values in Script 2 are:
Berlin
```

```
Munich
The values at the end of Script 1 are:
Berlin
Los Angeles
```

Because variable1 is declared as permanent in both Script 1 and Script 2, it retains the value assigned to it in Script 2 even after Script1 has resumed execution. Variable2, however, is not declared as permanent in both scripts. Therefore the value it is assigned in Script 2 in not carried over when Script 1 continues.

Comments

Descriptive remarks, called comments, may be added to your script file at any point. A comment begins with a semi-colon (;).

```
; This script purges HP 3000 files specified by the user.
```

You cannot continue comments over multiple lines by using the ampersand character, as the ampersand will be ignored. Each comment line must begin with its own semi-colon.

```
;  Comments are extremely important for making you script easy
;  to read and understand.
```

Because MS92 ignores any characters on an input line once it has encountered a semi-colon (unless the semi-colon is in a quoted string – see below), you can also place a comment following a short command or statement on the same line:

```
 display variable1 + CR                ; Show the values of the
variables
 display variable2 + CR                ; the variables again
```

Strings

A string is any series of characters interpreted as text rather than numeric information. Most of the parameters of MS92's commands require string data. For example, the message command has the following syntax;

```
message [into variable] [prompt prompt_string]
[[icon] note|warn|stop] buttons string [ ,string [ ,string]]
```

For each string parameter in the command above, you may specify either a literal string enclosed in quotes, or an expression that yields a string. In the example below, the strings "No files found" and "OK" are displayed exactly as shown in the command line.

Message prompt "No files found." Icon warn buttons "OK"

Alternately, you could specify a variable which contains string data.

Myprompt# = "No files found."
Message prompt myprompt# icon warn buttons "OK"

This too displays "No files found." Another way of specifying a string parameter is to build an expression that yields a string. For example concatenated string expressions and chunking expressions (discussed below) both produce string data; so do some system-defined functions such as getfilename().

You cannot pass numeric data to a command parameter which requires string data, or the converse. Use the stringTo Number and numberToString functions to perform data type conversions.

All of TermTalk's system-defined terms – commands, logical operators, function names, configuration settings, and so on – can be entered in either upper or lower case. A complete list of these is found in Appendix A.

For quoted strings, case is significant. For example, if you compare the string "HELLO" to the string "hello," they are not considered equal. Lower case is evaluated as greater that upper case.

String concatenation

The + symbol can be used to assemble. Or concatenate, a number of string elements. For example. "hello" + cr 1f follows the string hello with a carriage return and a line feed. String concatenation is also used in the sample script:

sendline "LISTF " + filespec + ",2"

Here, an HP 3000 command string is built by putting together three elements:

- The string LISTF
- The contents of the variable filespec
- The string, 2

Thus if filespec contained @.PUB.PROGS, the string produced would be

LISTF @.PUB.PROGS,2

The elements of a concentrated string expression can include:

- Quoted strings
- Variables that hold strings
- System-defined functions that produce strings
- Control characters

To place spaces between elements, include the spaces within the quoted strings.

Special Characters in Quoted Strings

Both the caret symbol and the double quote symbol have special meanings when both used in quoted strings ---^ signifies a control character and " terminates the string. Thus there is a special way to enter these characters within a quoted string so they are transmitted literally. Precede them with a tilde, as follows:

~" ~^

Because of this convention, if you created a quoted string with a tilde as the last character, the closing quotes would be transmitted literally rather than interpreted as terminating the quoted text. Therefore, to specify a literal tilde at the end of a quoted string, use two tildes in a row:

~~

For example, "abcd~~" produces the string abdc~.

A semicolon used in a quoted string is transmitted literally; it is not interpreted as the beginning of a comment.

Since a quoted string cannot span multiple lines, its maximum length is the maximum number of characters in an input line allowed by the text editor you use to create the script file, up to a limit of 255 characters.

Chunking

Chunking is a way of specifying part of a string to be extracted from the whole – similar to what is called "substringing" in some other computer languages.

For example, chunking is used in this portion of the sample script to extract the file names from the directory listing produced by the previous LISTF , 2 command, now stored in the variable list_of_files. LIST, 2 produces a listing that includes one line per file. The first word of each line is the file name itself. Each file name is placed in the variable current_file, and sent to the HP 3000 to be purged.

```
I = 7
while I <= numberOfLines (list_of_files) and line I of &
     list_of_files <> “”
    current_file = word 1 of line I of list_of_files
    sendline “PURGE “ + current_file
    I = I + 1
endwhile
```

In the preceding example, the chunking keywords word and line are used to extract the desired portion of the listing.

There are four chunking keywords available:

Keyword	Meaning
character	Any ASCII character (abbreviated char)
word	A group of contiguous characters, including punctuation, delimited by spaces and / or end of line characters
item	A group of contiguous characters, including punctuation and spaces, delimited by commas
line	A group of contiguous characters, including punctuation and spaces, delimited by end of line characters (or, if the last line of the text does not include an end of line character, the end of the string)

These keywords are assembled using of:

```
char 5 of word 3 of “A quoted string like this”
```

This statement refers to the character n in the word string.

You can also specify a range of elements, using to:

```
char 2 to 5 of word 3 of line 11 of output_text
```

This statement refers to four characters in the third word of the eleventh line of the text stored in the variable output_text.

Functions Used with Strings

The following system-defined functions operate on strings. Complete syntax of TermTalk’s system-defined functions is found in Chapter 4, page 58.

Function Name	Purpose
find	Searches for substring in source string
lower	Converts uppercase to lower
numberOfChars	Returns number of characters in string
numberOfItems	Returns number of items in string
numberOfLines	Returns number of lines in string
numberOfWords	Returns number of word in string
numberToString	Converts numeric expression to string
stringToNumber	Converts string to numeric expression
upper	Converts lowercase to upper

For example, in this section of the sample scrip, the numberOfLines function is used:

```
I = 7
while I <= numberOfLines (list_of_files) and line I of
list_of_files <> “”
    current_file = word 1 of line I of list_of_files
    sendline “PURGE” + current_file
    I = I + 1
endwhile
```

The numberOfLines function is performed on the argument list_of_files to determine how many lines there are in the list of files produced by the LISTF, 2 command. The result is the total number of individual files that have to be purged. This information is used to control the duration of the operation which purges the files one by one.

Empty String

“” (a pair of quotes with nothing between them) denotes an empty string. Testing for an empty string is useful to determine, for example, whether the user has entered data in response to a prompt.

```
; These commands display a dialog box in which the user enters a file
; name. If none is entered, the script terminates.
FileName# = getFileName (“Select file to display:”, “*.*)
if fileName# = “”
```

```
        stop
    endif
```

A box like this is displayed:
[p. 29]
If the user does not enter a file name, the variable contains an empty string. The stop command terminates the script.

Numeric data and arithmetic expressions

Some of the parameters of MS92's commands require numeric data, These parameters are shown in italics in the command syntax provided in Chapter 4. For example:
tab [at colum_number]
When numeric data is required, you must either supply a number or an arithmetic expression – a sequence of operators and operands that computes a numeric value. For example, all of the following yield numeric values.
I + 1
baudrate/100
(5 + 3) / (x - 1)
5 (This is a numeric constant)
The following arithmetic operators can be used:

Operator	Meaning
+	Addition (also used to concatenate strings, as explained in the "Strings" section on page 25)
-	Subtraction (if used with single operand, reverses the sign of the operand)
*	Multiplication
/	Division
MOD	Modulo (integral remainder from division)

The operands used in an arithmetic expression can be any of the following:

- Signed integers ranging from - 2147483647 to 2147483674.
- Variables that hold numbers.
- System-defined functions that produce numbers.
- Arithmetic expressions.

Note that floating point arithmetic is not available, and real numbers (numbers with fractional parts) are not supported.

Normally, expressions are evaluated from left to right. Multiplication, division, and modulo operations precede addition and subtraction.

Parentheses can be used to control order of operations, with the expressions in the innermost parentheses evaluated first.

The result of the evaluation of a particular arithmetic expression may be assigned to a variable with the = symbol. The variable is always specified on the left side of the = sign, the expression on the right. For example:

```
I = 0
I = I + 1
get baud into x
x = x/100
value = ((5 + 3)*100)/(x - 1)
```

You cannot pass numeric data to a command which requires string data, or the converse. Use the stringToNumber and numberToString functions to perform data type conversions.

Logic and flow control

TermTalk provides a number of statements which control what parts of a script are performed and in what order. These are shown in the sample script below.

```
Label get_input
input into answer prompt “Enter Y or N: “
if answer = “Y”
; Do appropriate actions
elseif answer = “N”
; Do appropriate actions
else
goto get_input
endif
```

In this script, the input command prompts the user to enter Y or N. If the user answers Y, one set of actions is performed. If the user enters some response other that Y or N, the prompt is displayed again. To do this, the following keywords are used.

Keyword	**Purpose**
if condition	Sets condition. Followed by action to be taken.
elseif condition	Sets another condition. Followed by action to be taken.
else	Action to be taken if neither if nor elseif is true.
endif	Ends the current if.
label label_name	Specifies a name which identifies a particular part of the script, for use with goto.
goto label_name	Transfers processing to the part of the script identified by label_name.

Also useful in setting up logical conditions to control the flow of your script is the while keyword. In the sample script at the beginning of this chapter, while is used to establish a loop:

```
I = 7
while I <= numberOfLines (list_of_files) and line &
    I of list_of_files <> “”
    current_file = word 1 of line I of list_of_files
    sendline “PURGE” +current_file
    I = I + 1
endwhile
```

Here the variable I is established to track progress through the lines of data contained in list_of_files. As long as I is less than or equal to the number of lines in list_of_files, and if the current line is not blank, the file on the line pointed to by I is purged. Once I is equal to the number of lines in list_of_files, the operation is complete.

Keyword	**Purpose**
while	Sets a statement or a series of statements to be executed repeatedly (a loop) until the controlling logical expression is false.
endwhile	Ends the current while.

The relational operators listed on the following page are useful in developing logical expressions. If two expressions are joined by the operator on the left, the resulting expression is true if the condition on the right exists. If not, the expression is false. False is defined to be the value zero; true is any non-zero value.

Operator	True if
<	First expression is less than second expression.
>	First expression is greater than second expression.
<=	First expression is less than or equal to second expression.
>=	First expression is greater than or equal to second expression.
<>	First expression is not equal to second expression.

Operator	True if
=	First expression is equal to second expression.
Contains	Second string is a substring of the first string.
And	First and second expression are both true.
Or	First or second or both expressions are true.
Not	The single expression to which it applies is false.

Note that you must compare numbers to numbers and strings to strings; you cannot compare arithmetic expressions to strings. (IF you do, you will get the compiler error telling you that the operator you specified must be applied to operands of the same type.)
When comparing strings to strings, lower case is considered greater than uppercase, and a longer string is considered greater that a shorter one.
The following additional commands can also be used to control the flow of a TermTalk script. These are fully documented in Chapter 4, *Commands and Functions*, on page ???.

Command	Purpose
do	Performs a user-defined procedure, established with proc statement.

Do script	Performs the script in the specified file.
Return	Terminates execution of current procedure or current script. If the current script was initiated by another script, returns control to initiating script.
Stop	Terminates execution of all currently-active scripts.

System-defined functions

A function is a system-defined procedure that performs an operation and returns a value. The argument, or target of the function's operation, is always specified in parentheses. TermTalk's functions include:

Type of Function	**Examples**
String Functions	find(), lower(), numberOfLines()
Variable Functions	isString(), isNumber(), isEmpty()
System Functions	freeDisk(), identify(), connected()
File Functions	newFileName(), getFileName()
Error Handling Functions	error(), errorLine(), errorString()

The syntax of each TermTalk function is described in detail in the alphabetical reference in Chapter 4.

In the following example, the numberOfLines function is used.

```
I = 7
while I <= numberOfLines (list_of_files) and line &
    I of list_of_files <> ""
    current_file = word 1 of line I of list_of_files
    sendline "PURGE" +current_file
    I = I + 1
endwhile
```

The numberOfLines function is performed on the argument list_of_files to determine how many lines there are in the list of files produced by the LISTF,2 command. This information is used to control the duration of the operation which purges the files one by one.

User-defined procedures

A user-defined procedure is a portion of the script that can be performed repeatedly without duplicating any code. The procedure is preceded and named by the proc statement and concluded with the endproc statement. To perform the procedure, use the do command and specify the procedure name established with proc.
The use of these statements is shown in the example below.

```
Proc saveSettings
    get baud into oldBaud
    get hostPrompt into oldPrompt
endproc
proc restoreSettings
    set baud to oldBaud
    set hostPrompt to oldPrompt
endproc
do saveSettings
set baud to 9600
set hostPrompt to ":^Q"
sendline "hello manager.sys"
sendline "run myprog"
sendline "bye"
do restoreSettings
```

User-defined procedures must appear at the beginning of the script file, immediately following the declaration of any permanent variables.
Make sure the name you choose for your procedure does not conflict with any label name or variable name you are using in the same script, or with any system-defined term listed in Appendix A. A procedure name may be terminated with a # symbol to distinguish it from a similar label, variable, or system-defined term.

Error handling

Two types of errors can occur during the execution of a TermTalk script: fatal and nonfatal. If a fatal error occurs during the execution of a script – for example, arithmetic overflow or division by zero – the error is reported and all currently-active scripts are terminated.
If a non-fatal error occurs, MS92's default is to report the error and terminate execution of all currently active scripts. However, you can override this default by specifying the ignore errors command. To return to default operation, specify trap errors. If scripts have been nested, ignore errors affects only the currently-executing script.
TermTalk also provides a number of functions which control error handling. These functions are only useful after an ignore errors statement, since otherwise errors cause termination of the current script. These functions include:

Function	Purpose
error()	Monitors error status of previous command: returns error number zero for "no errors".
errorline()	Returns source file name and line number where last error occurred.
errorstring(expression)	Returns error message corresponding to the error number specified by the expression, To retrieve the error message for previous error, specify errorstring(error()).

The following scripts show some error handling techniques.

```
;Script One
ignore errors                ;Do not quit if an error occurs
open file "MYFILE"
if error()                   ;If error occurs on open file
create file "MYFILE"         ;file doesn't exist. Try to
;create it.
If error()              ;if this fails, show line#
;and msg.
Display errorline() +errorstring(error())
stop
endif
```

```
endif
trap errors          ;monitor errors again
Script Two
ignore errors              ;Do not quit if an error occurs
open file “MYFILE”
if error()                 ;If error occurs on open file
create file “MYFILE”       ;file doesn’t exist. Try to
    ;create it.
If error() = 203           ;check for the error
    ; “FileExists”
display “That file already exist.”
Stop
endif
endif
trap errors          ;Monitor errors again
```

Chapter 3
What Commands Can Do

Introduction

This chapter groups TermTalk's scripting commands and system-defined functions in terms of the tasks they are used to perform. These tasks include:

- Communicating with the User
- Setting and Retrieving Configuration Values
- Communicating with the Host
- File Transfer
- Working with Files
- Controlling MS92
- Editing Data on the Screen

In each section, a list of commands relating to the task is provided, along with an example illustrating how they are used. Most of these examples illustrate other commands and techniques as well. Going over them is good way to familiarize yourself with TermTalk.
Refer to Chapter 4 for complete details on each command, along with additional examples.

Communicating with the user

These commands are used to exchange information with the person who is running the script.

Command	Purpose
input	Displays a dialog box in which user can enter a text string.
message	Displays a message box.
Display	Displays a string or the result of an expression
beep	Produces an audible tone.

Also useful are the functions, getFileName() and newFileName(), which display dialog boxes in which the user can choose or create a file, respectively. The prompt you specify is displayed in the bar across the top of the box, as shown in this sample getFileName dialog.
[p. 40 (2)]

Example

This example uses he input, beep, and message commands, as well as sendline.input is used to prompt the user for his logon and password. Beep is used after logging on to get the user's attention, then a simple message is displayed with the message command.

```
; Logging on to the HP 3000
; Prompt the user for his logon string with the input command
; Test for empty string, returned if the user clicks Cancel
input into logOnString prompt "Enter you Logon:"
if lofOfString = ""
    stop
endif
'Prompt the user for his password string with the input command
input password into passwordString prompt "Enter you password:"
;Test for empty string, returned if the user clicks Cancel
if passwordString = ""
    stop
endif
'Send logon to the HP 3000 and wait for prompt
```

```
sendline logOnString
expect ":" + dcl
; Send the password to the HP 3000 and wait for the prompt
sendline passwordString
expect LF+":" + dcl
; Tell the user he or she is logged on
beep
message prompt "Logged on." Buttons "OK"
```

The first input command produces a dialog box like this:
[p. 41 (1)]
The second input command produces a dialog box like this:
[p. 41 (2)]
The message command produces a dialog box like this:
[p. 41 (3)]

Setting and retrieving configuration values

These commands are used to manipulate host configuration values.

Command	Purpose
set	Changes a configuration setting to the specified value.
get	Retrieves one of MS92's configuration settings into the specified variable.
setup	Creates a dialog box to allow the user to change MS92's configuration.
open session	Opens a MS92 configuration file (a "settings file") and sets the configuration options contained therein.
save session	Saves the current configuration to a settings file.
revert	Returns all configuration settings to the values saved in the current settings file, or if no settings file is open, to MS92's default values.

Example

MS92's set and get commands are used to specify and retrieve the contents of a variety of system-defined configuration settings. For example, the setting fontsize can be set to any value supported by MS92. A complete list of configuration settings can be found in Chapter 4. The script on the next page specifies communications and terminal parameters as well as some screen display elements. Settings are then saved to a settings file. In the section of the script labeled "Terminal Settings," notice that yes, on, and true can be used interchangeably when setting configuration parameters, as can no, off, and false.

```
;Configuring the Terminal and the Screen
;Communications settings
set hostPrompt to ":" + dcl
set baud to 9600
set dataparityBits to eightNone
;Terminal settings
set display to yes
set smoothScroll to no
set enqAck to on
set typeAhead to off
set autoLineFeed to true
set blockMode to false
; Function key settings
set f1 label to "log on"
set f1 message to "hello manager.sys"
set f8 label to "log off"
set f8 message to "bye"
; Screen settings
set text window location to "30,40"
set text window title to "HP 3000 Defaults"
; save all settings to the MS92 Defaults file
save session "HP3000.SCF"
; the settings file could be opened like this
open session "HP 3000.SCF"
; Here, the baud setting is retrieved using the get command,
; set to a new rate, then reset to the original value
get baud into originalBaudRate#
set baud to 1200
set baud to originalBaudRate#
```

Communicating with the host

These commands are used to interact with the host computer.

Command	Purpose
break	Sends a break signal to the host computer.
connect	Connects to a host computer.
disconnect	Terminates the connection to the current host computer
dial	Dials a phone number through an attached modem.
sendline	Sends a string plus a carriage return to the host computer.
sendstring	Sends a string, without a carriage return to the host computer.
expect	Receives data from the host computer.
key	Simulates pressing various keys on the terminal keyboard.

Examples

The following sample script connects to an HP 3000 using a LAN, reads the current jobs running on that system into a file, then disconnects. It uses the connect, disconnect, sendline, and sendstring commands to do so. It also demonstrates how to use MS92 commands to create a file, open it, write to it, and close it.

```
; Communicating with the Host
;first ask the user for the name of the desired host
input into myHostName prompt “Enter the host for job listing:”
;Test for empty string, returned if user clicks Cancel
if myHostName = “”
    stop
endif
;disconnect from the host if already connected
disconnect
;now connect to the host specified by the user
connect myHostName
;log on to the host computer using another script called “logon”
```

```
do script “logon”
;issue the SHOWJOB command using sendline, then wait for prompt
;the job listing output from the host is stored into the
;variable joblist
sendline “showjob”
expect “:” + dcl into joblist
;save the job listing in a file whose name is created from the
;host name
jobListFileName = char 1 to 8 of myHostName + “.JOB”
;the jobListFileName now looks like “series70.JOB”
;create and open the file
create file jobListFileName
;write the job listing obtained aboved to the file
write jobList to file jobFileListName
;and close the file
close file jobListFileName
;log off the host
;since sendstring doesn’t append a carriage return for us, add
;one by concatenating cr to the end with +.
Sendstring “bye” + cr
;disconnect from the host
disconnect
```

The next script dials an HP 3000 using a modem, gets a prompt, and displays a message on the terminal screen. It uses the dial, break, and key commands, among others.

```
;Dialing In To A Host
;first dial the host computer using the modem
dial “555-1212”
;get the host’s attention using three returns
key return 3 times
expect “:” + dcl
;now display a message on the screen
;first home up and clear screen
cursor homeUp
clear screen
displayesc + “&db” + “Log in to your host machine:” + esc + “&d@”
```

File transfer

These commands are used to send and receive files with the host computer. Transfer protocol and file conversion options are specified as command parameters.

Command	Purpose
send	Sends a file to the host computer.
receive	Receives a file from the host computer.

Example

This sample script downloads a file on the host which contains a list of files to be backed up. It then transfers each of the files listed.

```
; Backing Up Files To the Host
;receive a file “download” from the host to the PC file
;”files.txt” as a text file, deleting it if it already exist
receive “download” to “files.txt” as text delete
; open the file just received
open file “files.txt”
;read the first line of the file into the variable currentFile
read file “files.txt” for 1 line into currentFile#
;loop through the list until currentFile is an empty string,
;meaning we have reached the end of the file
while currentFile# <> “”
    ;send file in currentFile to host, deleting it if already
    ;exist
    send word 1 of currentFile# to word 1 of currentFile# as backup &
    delete
    ; now read the next file to send from the file
    ; send a cr to the host to assure the host is ready for next file
    sendline””
    read file “files.txt” for 1 line into currentFile#
endwhile
;close the file list
close file “files.txt”
```

Working with files

These commands are used to manipulate files on your Macintosh.

Command	Purpose
copy file	Copies an existing disk file.
delete files	Deletes an existing disk file.
list files	Places a list of files into the supplied variable.
rename	Renames a file on disk.
close file	Closes a file opened with open file or create file.
create file	Creates and opens a new ASCII text file.
open file	Opens an ANSII text file.
print	Prints a text file, the graphics screen, or the current log.
read file	Reads from the current read file pointer of a disk file previously opened with open file or create file.
save	Saves MS92 data into a file.
write	Writes an expression to the write file pointer of a disk file previously opened with open file or create file.

Example

This sample script saves the current screen contents to a file and sends that file to the printer. This is a convenient way to print a copy of your screen without affecting any logging that may be in progress.

```
;Printing Data on this Screen
;First copy the screen contents to the variable: 'theScreen'
select screen
copy selection
paste into theScreen
;Create 'tempfile'. Write the contents of 'theScreen' to it,
;and close it.
```

```
Create file "tempfile"
write the Screen to file "tempfile"
close file "Tempfile"
;Now print the file on the printer
print file "tempfile"
;Delete the temporary file
delete file "tempfile"
```

Controlling MS92

These commands are used to control various elements of MS92s operation.

Screen control

Command	Purpose
maximize	Enlarges the specified window to full size of monitor.
Minimize	Collapses MS92 application to an icon.
Show	Makes the specified display element visible to the user.
Hide	Hides the specified display element from the user's view.
Restore	Restores MS92's windows to their normal size.

Script control

Command	Purpose
compile	Translates a source script into a compiled script.
do	Performs a user-defined procedure established with proc statement.
do script	Invokes the specified script.
return	Terminates execution of current procedure or current script. If the

	current script was initiating by another script, returns control to initiating script.
stop	Terminates execution of all currently-active scripts.
wait	Pauses script execution until a specified time.
ignore errors	Causes MS92 to ignore non-fatal errors, so that no message is displayed and the script continues execution.
trap errors	Reinstates error trapping, so that whenever an error is found, a message is displayed and the script terminates.

Miscellaneous

Command	**Purpose**
open	Opens an application.
page setup	Display a dialog box containing page setup options.
reset	Resets the terminal.
quit	Terminates execution of MS92.
log	Logs data in display memory. Logged data can be printed with print log, saved to a disk file with save log, copied to the Clipboard with copy log, or cleared with clear log.

Examples

This script sets the logDirection to top, so that lines are logged as they scroll out of display memory, then turns logging on. When logging is finished, the logged data is saved to a file.

```
;Logging Screen Data
set logDirection top
start logging
;To stop logging later on
stop logging
'To save the log file
save log "mylog"
```

This script automates routine tasks a user might run on a daily basis, freeing the user to take care of other business.

```
;Automating Routine Tasks
set transferProtocol to link3000
set hostPrompt to ":^Q"
do script "LOGON3000"              ;Run logon script
do script "PRINTMAIL"              ;Run script to print mail
ignore errors                      ;Do not quit if an error occurs
receive "BACKSTAT" delete          ;Transfer a file to PC
if error()
    result = "unsuccessful"
else
    result = "successful"
endif
trap errors                    ;Allow MS92 to monitor for
            ;errors again.
Sendline "BYE" ; Log off
; Display a message telling the user the job has finished
message prompt "Your daily routine is finished^M" &
    "Your mail has been sent to the printer^M buttons "OK"
quit
```

Editing data on the screen

These commands are used to modify data on the screen.

Terminal editing

Command	Purpose
delete char	Deletes character under the cursor and moves rest of the line one space left.
Delete line	Deletes line cursor is on and moves rest of display up one line.
Insert char	Switches between insert character mode and overstrike character

	mode.
Insert line	Inserts a line before the line the cursor is on and moves all lines below the new line down.
Cursor	Controls cursor movement.
Graph	Performs graphics functions.

PC editing

Command	Purpose
clear	Removes specified element.
Copy	Copies information to clipboard.
Cut	Places information on the clipboard and deletes the original.
Paste	Pastes the contents of the clipboard to display memory at the current cursor location or into the specified variable.
select	Selects or deselects parts of display memory.
tab	Sets a tab at the specified column.

Examples

This script allows the user to set a series of tab positions.

```
; Set a special group of tab settings:
MyTabSettings# = "7,8,12,16,20,24,28,32,36"
I = 1
while I <= numberOfItems (MyTabSettings#)
    thisTab# = stringToNumber (item I of MyTabSettings#)
    tab at thisTab#
    I = I +1
endwhile
```

This script allows the user to select a group of lines on the screen, delete them, and replace them with the contents of the clipboard.

```
;This script replaces a range of lines with the contents of
;the clipboard.
```

```
;Check Remote Mode status. Turn it off data won't be sent to host.
get remote into OrigRemote#
if OrigRemote = 1
    set remote to off
endif
;Check AutoLineFeed Status, set to on.
get AutoLineFeed in OrigLF#
if OrgLF = 0
    set AutoLineFeed to on
endif
input into deleteLines#
"Enter first and last line separated by a comma: "with "1,24"
; Check for cancel or empty string
if deleteLines# <> ""
firstLine# =stringTo Number (item 1 of deleteLine#)
if numberOfItems (deleteLines#) =2
    lastLine# = stringToNumber (item 2 of deleteLines#)
    select line firstLine# to lastLine#
else
    select linefirstLine#
endif
clear selection                 ;Remove data from selected lines
cursor firstLine#               ;Move cursor to first line
paste                           ;Insert data
endif
```

Chapter 4
Commands and Functions

Introduction

This chapter contains an alphabetical listing of commands, system-defined functions, and configuration settings.

Commands

Please take note of the following when using the command syntax in this section.

Platform-specific Commands

Some commands are platform specific, meaning that they are valid only under the Windows or Macintosh operating systems. Such commands are indicated by marginal boxes like the one shown at left.

Keywords

Keywords that must be entered exactly as shown are in boldface type. For example:

copy file old_name **to** new_name [**delete**]

User-Specified Parameters

Parameters for which you substitute the value of your choice are in regular type, like both old_name and new_name in the example above.

String and Numeric Data

Command parameters that require string data are shown in normal, non-italic type, like script_name in the following example.

do script script_name

Parameters that require numeric data are in italics, like row_number in the following example.

cursor *row_number*[, *colum_number*] [relative]

When substituting values for command parameters, you may specify either an expression which evaluates to that value, or the value itself. If the value is string literal, it should be enclosed in quotes. For example, the do script command has the syntax:

do script script_name

If you specify "myscript" in quotes, it is considered to be the literal name of the script file.

do script "myscript"

If you specify myscript without quotes, it is considered to be a variable containing the script name.

do script myscript

You could also specify an expression which would return a file name. For example,

do script getfilename ("Which script?"."*.*).

You cannot pass numeric data to a parameter which is expecting string data, or the converse. Use the stringToNumber and numberToString functions to perform data type conversions on the command line.

For example, in the script shown below, stringToNumber is used to convert string data to the numeric data required by the tab command.

```
;Set a special group of tab settings:
MyTabSettings# = "7,8,12,16,20,24,28,32,36"
I = 1
while I <= numberOfItems (MyTabSettings#)
thisTab# = stringToNumber (item I of MyTabSettings#)
tab at thisTab#
I = I + 1
endwhile
```

Optional items

Optional items are surrounded by brackets. For example:

[as id_number]

Choose one

A group of possible items from which you must choose one is separated by vertical slashes. For example:

command_name | script_name | string_to_send

On and off

Note that true, yes, or any non-zero numeric value are equivalent to on. False, no, or 0 (zero) are equivalent to off. Thus insert char on is the same as insert char yes, or insert char true, or insert character 1.

Related settings

The operation of some commands is affected by the setting of certain configuration parameters. For example, the send command is affected by the value of the transferProtocol configuration settings. Under each command where applicable, a list of related configuration settings is provided. The configuration settings themselves are described following the alphabetical command listing in this chapter.

Errors

For each command which returns error messages, a list of possible errors and error numbers is given. Error numbers can be tested with the error() function, as shown in the following example:

```
ignore errors ; Do not quit if an error occurs
open file "MYFILE"
if error()                          ; If error occurs on open file
create file "MYFILE"                ; file doesn't exist. Try to
        ; create it.
If error() = 203                    ; Check for the error
        ; "FileExists"
display "That file already exist."
Stop
endif
endif
trap errors                 ; Monitor errors again
```

For more information

In many cases, the operation of TermTalk commands which perform terminal emulation functions overlaps with that of MS92's menu and dialog box items. For example, the various forms of TermTalk's log command provide the same functionality as commands on MS92's Log menu. Although all of these terminal emulation functions are described in this manual, in some cases more detail can be found by looking up the corresponding menu command in the Technical Reference.

System-defined functions

System-defined functions perform an operation on an argument and return a value. The argument is always specified in parentheses. The following are listed alphabetically along with the commands in the following reference, and are identified by the box you see in the margin. MS92's functions include:

String functions

upper lower numberOfLines numberOfChars
numberOfItems numberOfwords numberToString
StringToNum find

Variable Functions

isString isNumeric isEmpty

System Functions

freeDisk freeMem identity connected

File Functions

newFileName getFileName exist endOfFile

Error handling functions

error errorLine errorString

Configuration settings

Configuration parameters are managed with the set and get commands. They are indicated in the alphabetical reference by a box like the one shown at the left. The following illustrations show how the various configuration settings correspond to the options in MS92's dialog boxes. Note that the parameters must be preceded by a set or get command, so that the sessionName parameter must be entered as set sessionName.

The parameters listed below fall into three basic types. Some have a group of predefined acceptable values, or constants – logDirection, for example, can be set to either top or bottom; leftmargin must be set to a number form 1 to the column number of the right margin. These are called numeric parameters, because even though in some cases the acceptable values are words

rather than numbers, they are evaluated internally as numbers. If you attempt to set one of these parameters to a value other than those predefined, and error will result.

String parameters have no predefined values, and can contain any string which is appropriate – phoneNumber, for example, can be set to any sequence of characters you wish the modem to dial.

Boolean parameters are turned on and off – BlockMode, for example, or keyboardLock. A Boolean parameter must contain a numeric value, or a constant which evaluates to a numeric value. Constants include yes, on, and true (evaluate to 1) and no, off and false (evaluate to 0). Any non-zero numeric value is equivalent to true.

Not every configuration setting corresponds to an option in the user interface, but nearly all of them do, as indicated by the dialog boxes below.

[p. 63 (1)]

The terminal Settings dialog box (select **Terminal** from MS92's Settings menu.)

[p. 63 (2)]

Alphabetical listing

agentTaskRunning ()

```
;If the task is already running, stop
if agentTaskRunning ()
    stop
;Otherwise, start the task
else
    do agent task "mytask"
endif
```

alternateSet

The current alternate character set. When the emulation parameter is set to hpText, possible values are normal, and linedrawing. When the emulation parameter is set to vt100 or hpAnsi, possible values are normal, lineDrawing, and uk.

autoHorizScroll

A boolean parameter which, when set on, enables automatic horizontal scrolling. The default value is off.

autoKeyboardLock

A boolean parameter which, when set on, enables automatic keyboard lock. The default value is off.

autoLineFeed

A boolean parameter which, when set on, enables the automatic linefeed option. The default value is off.

backspaceKeyResult

The current function of the Backspace key. Possible values are backspaceKey, which performs a backspace without deleting the character to the left of the cursor, destructiveBackspace, which performs a backspace followed by a space and then another backspace, deleting the character to the left of the cursor, or delCharacter, which is utilized by some applications.

Baud

The current baud rate. Possible values are 300, 600, 1200, 2400, 4800, 9600, and 19200. The default value is 9600.

Beep

Produces and audible tone

```
beep
```

blockMode

A boolean parameter which, when set on, enables the block mode strap. The default value is off.

blocksize

The block size in bytes used during file transfer. Can be set to any value from 1 to 1024. Note that if you are transferring a file from a floppy disk on the PC to a host over a 2400 baud connection, you may need to lower the blocksize to 512 to compensate for the slowness of the PC floppy disk drive.

Break

Sends a break signal to the host computer, temporarily interrupting PC / host communications.

```
Break [length_of_signal]
```

Note length_of_signal is specified in milliseconds; the maximum length is 5000 milliseconds. The default is 100 milliseconds.

```
;Send a 300 millisecond break on the connection
break 300
```

Clear

Removes specified element

clear line	Clears from cursor position to the end of the line.
clear selection	Clears current selection, if any.
clear screen	Clears text screen from cursor position to end of screen.

clear all	Clears display memory.
clear graph	Clears graphics memory.
clear log	Clears current log.
clear margins	Clears left and right margin settings; equivalent to setting the left margin to 1 and setting the right margin to the value of the columns configuration setting.
clear tab	Clears tab set at the current column, if any.
clear all tabs	Clears all tab settings.

Close file

Closes a file previously opened with open file or create file.

close file file_name

Note: file_name must exactly match the name specified with open file or create file.

```
;Open a file, write a message, and close the file
open file “TESTFILE.TXT”
write “My Message” to file “TESTFILE.TXT”
close file “TESTFILE.TXT”
```

Related Settings

directory

Errors

200	filePermissionError
201	fileNorFound
208	fileNotOpen

closeScript

The name of the script performed automatically when the settings file is closed.

Compile

Translates a source script into a commands script.

Compile source_file as compiled_file [password password_string]

Where password_string can be used to decrypt the source file if encrypted. If the password is omitted and the script is encrypted, the user is prompted for the password. If the script is not encrypted, the password is ignored.

MS92 recognizes the extension .TTS for source script files, and the extention .TTX for compiled script files.

```
; Compile a source file called mysource into file called
; myobject
compile “mysource.tts” as “myobject.ttx”
```

Related Settings

directory

Errors

200	filePermissionError
201	fileNotFound
202	fileIOError
203	fielExist
204	tooManyOpenFiles
205	fileBusy
206	fileTooLarge
207	diskFull
100	insufficientMemory

connect

Connects to a host computer.

Connect [name_of_host]

where name_of_host specifies the node name of the host computer. If name_of_host is not specified, connection is made to the host specified by the hostName configuration setting, which is initially hp3000, but may be changed with the set command.

When using a serial port, connect is ignored.

```
;Connect to a host selected by the user
input into newHost# prompt “Enter new host name:”
connect newHost#
```

Related Settings

baud
connection
hostName
modemType
phoneNumber
phoneType
port

Errors

250 connectFailed

compressXfer

A boolean parameter which, when set on, enables use of compression during file transfer. The default value is off.

connection

Possible values are: Serial, Modem, NSVT and Telnet.

copy

Copies information from a specified source to the clipboard
copy [selection] [as table]
copy graph [scaled]
copy windowImage
copy log
Use the select command to mark a screen selection from a script.
Specifying copy with no further parameters does the same thing as copy selection: it copies the current selection to the clipboard. If there is no current selection, the command does nothing. The keyword selection is only specified to improve the readability of your scripts.
A selection may be copied as a table.
If copy graph is specified, the contents of the graphics window are copied to the clipboard. If scaled is specified, the contents are copied as scaled to fit in the window.

If copy windowImage is specified, a bitmap of the current window is copied to the clipboard.
If copy log is specified, the contents of the current log are copied to the clipboard.

```
; Copy the current selection from the screen to the clipboard
copy
; Copy a bitmap of the current window to the clipboard copy
windowImage
```

Errors

100	insufficientMemory
200	filePermissionError (copy log only)
201	filenotFould (copy log only)
202	fileIOError (copy log only)
204	tooManyOpenFiles (copy log only)
205	fileBusy (copy log only)
391	errorAccessing Clipboard

connected()

Returns TRUE if a successful connection has been made with the remote host.

```
if connected ()
message prompt “Connected” icon note buttons “OK”
else
message prompt “Not connected” icon note buttons “OK”
endif
```

copy file

Copies an existing disk file.

copy file old_name to new_name [delete]

old_name is the name of an existing file; the file need not have been opened with the open file command.
new_name may be any file name which is valid by the rules of the operating system under which MS92 is running.
If new_name already exist and the delete keyword is specified, the file will be overwritten. If new_name exist and delete is not specified, a run time error occurs. If old_name does not exist or if the name supplied for new_name is invalid, a runtime error occurs.

```
;Copying files
; This script request the two files names involved in the copy
; procedure.
; The new file will overwrite any existing file of the same name
; without any notification.
Input into oldName# prompt “Enter source file name:”
input into newName# prompt “Enter target file name:”
copy file sourceFile# to targetFile# delete
```

Related Settings

directory

Errors

200 filePermissionError
201 fileNotFound
202 fileIOError
203 fileExist
204 tooManyOpenFiles
205 fileBusy
206 fileTooLarge
207 diskFull

create file

Creates and opens a new ASCII text file. Binary files are not supported.

```
Create file file_name
```

A fully-qualified file name can be specified, with or without a file extension. If not otherwise specified, files are created in the directory where MS92 is located, unless this default has been overridden by setting the directory configuration parameter or by changing the current directory in any file selection dialog box.

Two file pointers are maintained for the file, one for reading and one for writing. Reading from a newly created file returns and empty string.

```
; Create a new file
create file “TESTFILE.TXT”
```

Related Settings

directory

Errors

200 filePermissionError
203 fileExists
204 tooManyOpenFiles
205 fileBusy
207 diskFull

ctrlAltIsExtChar

A boolean parameter which, when set on, enables the use of the Control - Alt key combination to emulate the HP Extend Char key. The Extend Char key is used in combination with other keys to produce the HP Roman - 8 character set.

CurrentObject ()

Returns TRUE if there is a current object available for processing. A script invoked when an object is dropped into the MS92 window should not usually require this function.

```
;If a current object is available, send it to the host
if currentObject ()
    send object
```

cursor

Controls cursor movement and scrolling.
Cursor row_number[,column_number] [relative]
cursor up cursor down
cursor left cursor right
cursor rollRight cursor rollLeft
cursor rollUp cursor rollDown
cursor homeDown cursor homeUp
cursor prevPage cursor nextPage
In the first form of the command the cursor is moved to the screen coordinates specified. These coordinates are specified in terms of the whole of the display memory, not just the portion

visible on the screen. In other words, 1,1 is the first row and column in display memory, not the first row and column and the screen.
If column_number is omitted, a default value of one (1) is used.
If relative is specified, the cursor is moved the specific number of positions from the current location. Positive row and column numbers move down and to the right respectively; negative ones move up and to the left.
Cursor up, down, left and right move the cursor one character in the specified direction.
Cursor rollRight, rollLeft, rollUp and rollDown scroll the contents of the window one position in the specified direction. The commands are equivalent to the user clicking one of the arrows in the scroll bars one time.
Cursor homeDown moves the cursor to the left margin of the line after the last line in display memory. cursor homeUp moves the cursor to the left margin of the first line in memory.
cursor prevPage scrolls the display so that lines preceding the current screen in display memory, up to and including the line immediately before the top line, are displayed. The cursor is positioned to the top lift position on the screen. For example, if the window is currently displaying 24 lines, the display is repainted so the previous 24 lines are displayed.
Cursor nextPage scrolls the display so that lines following the current screen in display memory, beginning with the line immediately following the last line on the screen, are displayed. The cursor is positioned to the top left position on the screen. For example, if the window is currently displaying 24 lines, the display is repainted so the next 24 lines are displayed.

```
; Move the cursor around the screen.

cursor 20,10                ;Moves cursor to row 20, column      ;10
cursor 2,2 relative  ;Moves cursor 2 columns right
                            ;and 2 rows down
cursor homeUp               ;Moves cursor to Home (1,1)
cursor right                ;Moves cursor one space to the
                  ;right
cursor rollRight            ;Scrolls contents of window one
                  ;space to the right
cursor prevPage             ;Scrolls contents of window up
                  ;one “page” (24 lines)
```

Related Settings

memoryLock
numberOfColumns
numberOfScreens

leftMargin
rightMargin
xmitFunctions

cursorType

The shape of the cursor. The possible values are blockCursor and lineCursor. The default value is lineCursor.

Cut

Copies information from a specified source to a specified destination and deletes the source.

cut [selection]

Use the select command to mark a screen selection from a script.

selection may be specified to improve the readability of your scripts; however, the command always copies the current selection. If there is no current selection, the command does nothing.

;Cut from selected area of screen to clipboard:

cut

Errors

391 errorAccessingClipboard

100 insufficentMemory

dataParityBitOs

The number of data bits used in serial datacom. Possible values are eightNone, seveneven, sevenOdd, sevenOnes, and sevenZeroes. The default value is eightNone.

Date ( )

Returns the current data in mm-dd-yyyy format.

Display “The current date is: “+date ( ) + cr + lf

dc1Pacing

A boolean parameter which can be set on to enable DC1 pacing during file transfer to HP 3000 systems. The default value is off.

DefaultBinaryXfer

A boolean parameter which, when set on, enables the transfer of files in binary format. When off, files are transferred in backup format. The default value is off.

delayAmount

The amount of time in milliseconds paused after an escape character is sent to the host. Possible values are from 0 - 3. The default value is 0.

delete file

Deletes an existing disk file
delete file file_name
If the specified file does not exist, a runtime error occurs.
;Delete the file Testfile
delete file "Testfile"

Related Settings

directory

Errors

200	filePermissionError
201	fileNotFound
205	fileBusy

delete char

Deletes the character under the cursor and moves the rest of the line one space to the left.
Delete char

delete line

Deletes the line the cursor is on and moves the rest of the lines on the screen up one line.
delete line

dial

Dials a phone number through an attached modem.
Dial [phone_number_string] [with modem_commands]
If no phone_number_string is specified, the value of the phoneNumber configuration setting is used.
Any modem_commands specified are passed to the modem before the number is dialed, overriding MS92's default modem parameters. (See the Technical Reference for a list of the commands sent to the modem by default.)

```
;Send command to set modem's escape guard time
;with the phone number
dial "5124790735" with "ATS12=4"
```

Related Settings

baud
modemType
phoneNumber
phoneType
redial

Errors

260 modemNotResponding

directory

The name of the folder where files are accessed. The default is the name of the folder in which MS92 is located.

disableBell

A boolean parameter which, when set on, causes escape sequences for host control to be ignored. The default value is off.

disconnect

Terminates the connection to the current host.

disconnect

Related Settings

hardHangup

display

Displays the result of an expression on the terminal screen.

```
display expression
display file file_name
```

if an expression is specified, it may contain string or numeric data. The data is converted to a string before it is displayed.

file_name must be a Mac-based file. When a file is specified, its contense are displayed on the terminal screen.

```
display "myfile.txt"            ;Displays the string myfile.txt
display myfile                  ;Displays the contents of the
                                ;variable myfile
display file "myfile.txt"       ;Displays the contents of the ;
                                ;file myfile.txt
display 3 + 2 ; Displays 5
display date ( )                ;Displays the current system
                                ;date
display time ( )                ;Displays the current system
                                ;time
;To display "Message" with inverse video display enhancement
display esc + "&dB" + Message
```

Errors

For display file only:

200 filePermissionError
201 fileNotFound
204 tooManyOpenFiles
202 fileIOError
205 fileBusy

Related Settings

alternateSet
autoHorizScroll
directory (For display file only)
display
displayFunctions
fontSize
formatMode

display

A boolean parameter which, when set off, inhibits the display of output to the screen. The default value is on.

displayFunctions

A boolean parameter which, when set on, causes control characters to be displayed on the screen. The default value is off.

do

Performs a user-defined procedure, established with proc statement.
Do procedure_name
For information on proc statements, see page 36.

```
Proc saveSetting
    get baud into oldBaud
    get hostPrompt into oldPrompt
endproc
do saveSettings
```

draftprintSize

The current draft print size. Possible values are normal, compressed, and expanded. The default value is normal.

DropList ( )

The dropList ( ) function may be used in the dropScript to obtain a list of objects dropped onto MS92. The files and directories that were dropped are returned in a string separated by spaces; directory names have a backslash appended to them. You may use TermTalk chunking statements (see page 27) to access individual items in the list. If dropList ( ) is used in a script that is not the dropScript, it returns an empty string. For example:

```
;Display each of the items in the drop list one at a time
list# = dropList ( )
I = 1
while I <= numberOfLines (list#)
    display line I of line#
    I = I +1
endwhile
```

dropScript

The script to be executed when a file, a directory, or a group of file and / or directories is dropped into the MS92s window. This script is responsible for taking the appropriate action for each file or directory dropped. The script may obtain a list of the dropped objects by calling the droplist ( ) function. The default value is “”, the empty string when dropscript is set to “”, MS92 performs its default drag and drop behavior.

Emulation

the current type of terminal emulation. Possible values are hpText, hpAnsi, and vt100.

Encrypt

Encrypts a TermTalk script file or any other type of file
encrypt plain_file as encrypted_file password password_string

endOfFile (filename)

Returns TRUE if a file opened with the createfile or openfile command is closed or at the end of file.

```
open file “myfile”
while not endOfFile (“myfile”)
    read file “myfile” into line#
            display line#
endwhile
```

enqAck

A boolean parameter which, when set on, enables HP ENQ / ACK handshaking. The default value is on.

enterKeyResult

The current function of the Enter key. Possible values are returnKey and enterKey

error ( )

Returns 0 if no nonfatal errors have occurred in the previous command. Otherwise the error number is returned.
If an ignore errors command is issued so that MS92 does not automatically terminate after a non-fatal error occurs, the error status is updated to reflect the success of each command after execution. The error function can be used to monitor the status. If no ignoreerrors commands is issued, this function always returns 0.

```
Ignore errors
send “myfile”
if error ()
```

```
        message prompt srrorString (error()) buttons "OK"
        display "Error in file: " + errorLine()
    endif
```

errorLine ( )

Returns a string indicating the source file and a line number of the statement that produced that last recoverable error. If no errors have been encountered, this function returns an empty string. See example under error ().

errorString (error_number)

Returns a the last text description corresponding to the error number specified. If the number given does not correspond to a non-fatal error, an empty string is returned. To retrieve the error message for the last error that occurred, specify errorstring(error()).
See example under error().

exist (filename)

Returns TRUE if the specified file exist.
If exist (c:\autoexec.bat")
copy file "c:\autoexec.bat" to "c:\autoexec.bak"
endif

expect

Receives data from the host computer.
Expect expected_string
[into variable] [for number_of_units chunking_until]] [timespec]
[interactive]
where chunking_unit is one of the following:
character[s] | char[s] | line[s]
where timespec is one of the following:

wait number_of_units time_unit
wait until time_string
where time_unit is one of the following:
second[s] | minute[s] | hour[s]
where the format of time_string is:
“HH:MM”
“HH:MM:SS”
expected_string specifies a string to be received from the host. (As with all string parameters, you may specify either a string literal enclosed in quotes or the name of a variable containing the string.) The command terminates upon receipt of this string.
The string received from the host can be stored into a variable by specifying a variable name after the into parameter. The text received may be limited to a specific number of characters or lines with the for parameter.
timespec specifies how long to wait for the expected string. It can be specified to either as a number of seconds, minutes, or hours, or as a time of day (use 24-hour military clock). If no timespec is given, the number of seconds specified in the timeout configuration setting (one minute) is used.
Normally the user cannot interact with the terminal while the expect command is waiting for a string. If the interactive keyword is specified, however, characters typed during the waiting period are sent to the host.

```
;Receive LISTF data from HP 3000 and display file names
;on screen
sendline “listf @, 2”
expect “:^Q” into listOfFiles# for 20 lines wait 10 seconds
I=1
while I <= numberOfLines (listOfFiles#)
    currentFile# = word 1 of line I of listOfFIles#
    display currentFile# + lf + cr
    I = I + 1
endwhile
```

Related Settings

timeout

Errors

270 expectTimeout

find (substring, source_string, start_character_number)

Searches for an occurrence of a substring in a source string. If the substring is found, an integer represents its location in the source string is returned. If the substring is not found, 0 is returned. The first expression is the substring that is the target of the search. The second expression is the source string. The third expression is the location in the string is considered to be position 1.

```
TestData# = "abcdefgh"
foundStart# = find ("def", testData#,1)
if foundStart# <> 0
    display "Found!" + cr + lf
endif
```

fontSize

This is the size of the font used for output in the text window. Possible values are 6,7,8,9,10,12,14, and 16. The default value is 12. You can also use the value auto to enable scalable fonts.

FormatMode

A boolean parameter which, when set on, anables format mode. The default value is off.

freeDisk ( )

Returns the amount of free disk space (in bytes) on the currently logged drive.

```
DiskSpace# = freeDisk ()
display numberToString (diskSpace#) + "bytes available on disk"&
    + cr + lf"
memoryFree# = freeMem ()
display numberToString (memoryFree#) + "bytes free in memory "&
    + cr + lf"
```

freeMem ( )

Returns the amount of free memory space (in bytes). See example under freeDisk( ).

Get

Retrieves information about the session from which the current script was run into a variable.

get [the] setting into variable
get [the] [screen] row into variable
get [the] [screen] column into varaible
get [the] window_type [location | rectangle | title] into
variable
get [the] fkey fkey_descriptor into variable

where setting is one of the following: [???]

alternateSet	autoHorizscroll	autoKeyboardLock
autoLineFeed	backspaceKeyResult	baud
blockMode	blocksize	characterDelay
closeScript	compressXfer	connection
ctrlAltIsExtChar	cursorType	dataParityBits
dc1Pacing		defaultBinaryXfer
directory	disableBell	disableEnh
disableHostControl	display	displayFunctions
draftPrintSize	emulation	enqAck
enterKeyResult	escDelayAmount	fontSize
formatMode	formsCache	graphResolution
graphScaling	graphWindows	hardHangup
helpFile	hostFileStartup	hostName
hostPrompt	inhibitDc2	inhibitHandshake
inhibitLineWrap	keyboardLock	language
leftMargin	lineModify	localEcho
lockFkeys	logDirection	localEcho
lockFkeys	logdirection	maximizeSiz
memoryLock	modemType	NCSI
NCSIExtended	NCSIGeneralService	NCSIserver
NCSISpecific-Service		
	numberOfColumns	numberOfScreens
numericPlusKey-result		

	objectScript	openScript
pageMode	phoneNumber	phoneType
port	recode8bitXfer	recodeCtrlXfer
redial	remote	returnKeyResult
rightMargin	sessionName	shiftBackspace-Result
smoothScroll	spacesPerTab	tabKeyResult
terminalId	timeout	transferProtocal
transferTimeIncrement		
	typeAhead	waitHostPrompt
xcmodemBinary	xmitFunctions	xmodem1Kbinary
xmodem1KCRC16	xmodem1KGBinary	xmodemCRC16
xonXoffInput	xonXoffOutput	ymodem1KBlocks
ymodemGBinary	ymodsmBinary	zmodemCRC32
zmodemEncodeCtrls	zmodemFileMgmnt	zmodemFullPath
zmodemResume	zmodemTimeout	zmodemUpdateOnly
zmodemWindowSize		

where window_type is:
text | graphics | script
where fkey is:
f1| f2| f3| f4| f5| f6| f7| f8| f9| f10| f11| f12
where fkey_descriptor is:
label | message |script | attribute

The setting parameters allow you to retrieve the values of various configuration settings. For a complete description of the settings, see each setting's individual listing.

The row and column parameters allow you to retrieve the memory-relative position of the cursor. To retrieve the screen-relative position, add the screen keyword to the command.

The window parameters allow you to get the current location, rectangle, and title of each window. Location is returned as a string containing two comma-seperated numbers. Rectangle is returned as a string containing four comma-seperated numbers.

The fkey parameters allow you to get the current values of the label, message, script, or attribute associated with a specific function key.

```
;retrieve and store old baud rate prior to resetting it
get the baud into oldBaud#
set the baud to 1200
```

```
;do something like dial the modem, etc
set the baud to oldBaud# ; restores the original baud rate
get the formatMode into oldFormatMode#
;This time we'll do something with the information
get the numberOfScreens into oldNumberOfScreens#
display "The current number of screens = " &
    + oldNumberOfScreens# + lf + cr
get the fontSize into oldfontSize#
get the f1 label into f1label#
```

Errors

104 getCommandFailed

getFileName (prompt, qualifier)

Displays a standard Open file dialog box to allow the user to specify the name of a file to open, and returns the name of the file selected.
FileToSend# = getfileName ("select file to send:","*.*")
send fileToSend# as text
This dialog box would be displayed:
The qualifier is a four character string that represents the file type (PICT, WORD, and so on.) If the user selects cancel, an empty string is returned.

Graph

Performs the graphics functions listed below. Used only with graphic MS92 for Macintosh
graph black
graph cursor [on|off]
graph cursor x_coordinate, y_coordinate
graph pen x_coordinate, y_coordinate
graph pen to cursor
graph draw
graph rbline [on|off]
graph black makes the entire graphics window black.
Graph cursor with no parameters toggles the graphics cursor on and off.
Graph cursor followed by two coordinates moves the graphics cursor to the specified location.

Graph pen followed by two coordinates moves the graphics pen to the specified location.
Coordinates specified are absolute pixel coordinates.
Graph pen to cursor moves the graphics pen to the current cursor location.
Specify absolute pixel coordinates.
Graph draw draws a line from the pen to the cursor.
Graph rbline with no parameters toggles the rubberband line on and off.

```
graph cursor on
graph cursor 10, 20 moves graphics cursor to given coordinate
graph pen 30, 65 ;moves the graphics pen to given coordinate
graph draw ;draws a line from cursor to pen
```

Related Settings

graphResolution
graphScaling
graphWindows

graphResolution

The current resolution for the graphics window. Possible values are low, high, and hp264x. The default value is low.

GraphScaling

The current scaling used by the graphics window. Possible values are none, fit, and proportional. The default value is none.

GraphWindows

The number of windows used to display text and graphics. Text and graphics are displayed in the same window when this parameter is set to 1, or in separate windows when it is set to 2. The default value is 2.

hardHangup

A boolean parameter which, when set on, lowers the DTR line when hanging up the modem. The default value is off.

hostFileStartup

The string sent to the host to start the remote file transfer program. The default value depends on the value of the transferProtocal setting. If it is link3000, the default is RUN TYMLINK.PUB.SYS. If it is link9000, the default is /usr/bin/tymlink. If it is xmodem, no startup string is used.

hostName

The name of the host computer, used primarily for LAN connections.

HostPrompt

The prompt used by the host. MS92's default is ^Q, the DC1 character for HP 3000 machines. The correct value of this parameter is dependent on the configuration of each host computer.

Identity ( )

Returns a string representing the serial number and program version of MS92. The format of the string is as follows:

xxyyNVVVSSSSSSWWWtdzzz

xx	Platform the emulator is running on:	
	AM	Apple Macintosh
	MW	Microsoft windows
yy	Product number:	
	01	business
	02	Graphic

N	Product title
	1 MS92
	2 AdvanceLink
VVV	Version number
SSSSSS	MS92 serial number (not used with AdvanceLink)
WWW	Macintosh System or Windows version number
t	Whether TermTalk is available (Y/N)
d	Whether DDE is available (Y/N)
zzz	TermTalk version number

ignore errors

Causes MS92 to ignore non-fatal errors, so that no message is displayed and the script continues execution.

Ignore errors

If a non-fatal error occurs, MS92's default is to report the error and terminate execution of all currently active scripts. However, you can override this default by specifying the ignore errors command. To return to default operation, specify trap errors. If scripts have been nested, ignore errors is in effect only for the current script.

```
ignore errors                    ;do not quit if an error occurs
open file "MYFILE"
if error ()                ;If error occurs on open file
create file "MYFILE"             ;If file doesn't exist. Try to
                           ;create it.
If error () = 203          ;check for the error
                           ;"FileExists"
display "That file already exist."
Stop
endif
endif
trap errors                ;Monitor errors again
```

inhibitDc2

A boolean parameter which, when set on, inhibits the HP DC2 (H) strap. The default value is off.

inhibitHandshake

A boolean parameter which, when set on, inhibits the HP handshake (G) strap. The default value is off.

inhibitLineWrap

A boolean parameter which, when set on, inhibits the HP line wrap (C) strap. The default value is off.

input

Displays a dialog in which the user can enter a text string, and saves response under a specified variable name.

input [password] [into] variable [length length_of_input_string][prompt string] [with default_responce]

The input command opens a dialog box in the center of the screen. The box contains a prompt, if specified, a box for text entry, and OK and Cancel buttons.
If OK is clicked or if the RETURN key is pressed, the contents of the text entry box is placed into the specified variable. If Cancel is clicked, and empty string (“”) is placed into the variable.
password causes the text entered to be displayed as graphic characters for security purposes.
Length specifies the maximum length of the input text. If this parameter is omitted, up to 255 characters are accepted.
Any default_response specified is placed in the text entry box and is highlighted.
The user’s input is treated as string data. To use this data in a context that requires numeric data, the stringToNumber function must be used to convert the data type. See the last example for this command.

```
;Prompt for a file to be purged, specifying a maximum length
;of 8 characters
input into filename# length 8 prompt “enter file name to purg:”
sendline “purge “ + filename#
```

The preceding script displays the following box:

[p.100]

```
;Send a user-specified fileset for use in obtaining a file
;listing on the host:
defaultFileSet# = “@.@.ACCT”
input into fileSet# prompt “Enter file set desired:” with&
defaultFileSet#
sendline “listf “ + fileset#
```

The preceding script displays the following box:

[pg.101 (1)]

;Prompt user for a baud rate and convert to numeric type

;before setting baud

input into newBaud# prompt “Enter value for new baud rate:”

set the baud to stringToNumber (newBaud#)

The preceding script displays the following box:

[pg.101 (2)]

insert char

Switches between insert character mode and overstrike character mode.

insert char [on] [with wrap]

insert char off

If insert char is specified with no parameters, it toggles between insert character mode and overstrike character mode.

The with wrap parameter can be specified when turning insert character mode on. This causes characters to wrap to the next line when insertion causes the line to reach the right margin.

```
Insert char on                  ;Turns insert mode on
                                ;(it is off by default)
insert char off                 ;Turns it off
insert char with wrap           ;Turns it on with wrap
                                (“on” may be omitted)
insert char                     ;Toggles it off
```

```
insert char                         ;Toggles it on again
insert char on with wrap    ;Now wrap is on too
```

insert line

Inserts a line before the line the cursor is on and moves all lines below the new line down. The cursor is placed on the newly inserted line.

isEmpty (variable)

Returns TRUE if the variable does not contain a value.

```
FileName# = getFileName ("Select a file to send:","*.TXT")
if not isEmpty (fileName#)
send file fileName# as text
endif
```

isNumber (variable)

Returns TRUE if the variable contains a numeric value.

```
Variable1# = 10
if isNumber (variable1#)
display "Variable1 is a number." + cr + lf
elseif
display "Variable1 is a string." + cr + lf
endif
```

isString (variable)

Returns TRUE if the variable contains a string value.

```
Variable1# = "10"
if isString (variable1#)
display "Variable1 is a number." + cr + lf
else
display "Variable1 is a string." + cr + lf
endif
```

key

Equivalent to pressing the specified key the specified number of times.
Key [shift] key_name [number [times]]
where key_name is one of the following:

backtab	copyScreen	enter	PF1 - PF4
f1 - f12	lock	return	tab
select	stop		

shift is used only with function keys f1 - f12 to transmit the value of a function key when used in combination with the Shift key.
PF1 - PF4 are recognixed only when MS92 is emulating a VT100 terminal.
Number must be a number from 1 through 255. If this parameter is omitted, the keypress is simulated once.

```
;Log on to a host HP3000:
input into logonPhrase# prompt "Enter logon phrase:"
input password into myPassword# prompt "Enter password:"
key return 3 times ; get the attention of the host
sendline logonPhrase#
expect DC1
if numberOfChars(myPassword#) <> 0
sendline myPassword#
endif
;Simulate pressing of the f9 key
key f9
```

Related Settings
emulation
enterKeyResult
keyboardLock
returnKeyResult
spacesPerTab
tabKeyResult

keyboardLock

A boolean parameter which, when set on, locks the keyboard. The default value is off.

language

The language currently used by MS92 for Macintosh. Possible values are ascii7, ascii8, danish, french, german, norwegian, spanish, swedish, finnish, and uk. The default value is ascii7.

LeftMargin

The column position of the current left margin. Must be a number between 1 and the value of the rightMargin. The default value us 1.

LineModify

A boolean parameter which, when set on, enables line modify mode. The default value is off.

list files

Places a list of files inro the supplied variable.
list files [with filespec] into variable
The list produced includes each file name in the current directory or folder, one per line.
fielspec qualifies the file list with a file name and extention, such as *.TXT or *.*. MS-DOS wildcards? And * are both permitted.

```
;obtain a list of the files in the current directory
;and display them:
list files into fileList#
if numberOfChars (fileList#) <> 0
I = 1
while I <= numberOfLines(fileList#)
currentFile# = item 1 of line I of fileList#
display currentFile# + lf + cr
I = I + 1
get director into dirName#
display lf + “There are “ + I + “ files in the &
current directory (“ & + dirName# + “)” + lf + cr
endif
```

Related Settings
directory

localEcho

A boolean Parameter which enables the echoing of characters typed locally. The default value is off.

lockFkeys

A boolean parameter which, when set on, locks the function keys. The default value is off.

log

Logs various parts of a user's session.

```
log all
log line
log page
log selection
start logging
stop logging
log pagebreak
```

Logged data can be printed with the print log command, saved to a disk file with the save log command, copied to the Clipboard with the copy log command, or cleared with the clear log command.

log all logs everything in display memory.

log line logs the entire line that the cursor is on.

log page logs (from the cursor) the portion of display memory that is currently visible on the screen.

log selection logs the current selection, if any. Use the select command to mark a selection from your script.

start logging activates automatic logging. Every line is logged until a stop logging command is executed. The log direction (top and bottom) is specified with the logDirection configuration setting.

The stop logging command turns off automatic logging.

The log pagebreak command inserts a formfeed character (ASCII 12) into the current log.

```
set logDirection to top
start logging
```

```
;to stop logging later on
stop logging
;to save the log file
save log “logfile.txt”
```

Related Settings
log direction

logDirection

Determines whether lines are logged from the top or bottom or display memory. Possible values are top and bottom. The default value is bottom.

Lower (expression)

Converts upper case characters in the source string to lower case and returns the resulting string. Characters that do not belong to the set A-Z are unaffected. The source string is not altered by this function.

```
allower# = “abcedf”
allupper# = “ABCDEF”
lowerresult# = lower (allUpper#)
upperResult# = upper (allLower#)
if allLower# = lowerResult# and allUpper# = upperResult#
display “Successful!” + cr + lf
endif
```

maximize

Enlarges the specified window to occupy the entire monitor screen.
maximize [text|script]
text is the default.
Related Settings
maximizeSize

memoryLock

A boolean parameter which, when set on, enables memory lock. The default value is off.

message

Creates a message window with one to three buttons.

```
Message [into variable] [prompt_string] [[icon]
note | warn | stop] buttons string [,string[,string]]
```

If a prompt string is supplied, it is displayed in the window. The string can be no longer than 4 lines, with each line containing up to 30 characters. You must supply carriage returns between lines by using the ^M control character (within a quoted string) or CR (used outside the quotes).
If a variable is specified, it receives a value representing the button presses: 1 for the left most button, 2 for the center, 3 for the rightmost. The rightmost button will be highlighted for default selection with the Enter key.
If note, warn, or stop is specified, the corresponding icon is displayed in the window.
Button strings can be no longer than 9 characters. At least one button must be specified.

```
;Display a simple message
beep
message prompt “You are logged on.” icon warn buttons “OK”
```

The preceding script displays the following box:
[pg. 108 (1)]

```
;Test user’s response to a message
beep
message into response# prompt “Do you want to log off?” &
icon note buttons “Yes”, “No”
if response = 1
disconnect
endif
```

The preceding script displays the following box:
[pg. 108 (2)]

```
;Display a long message, including carriage returns
message into response# prompt “Do you wish to send another”&
+ CR + “file to the host?” buttons “Continue”,”Stop”
; Or you can do it this way
message into response# prompt &
```

```
"Do you wish to send another^Mfile to the host?" &
buttons "Continue","Stop"
```

Either of the message commands in the preceding script would display the following box: [pg.109]

modemType

The type of modem being used. Possible values are hayes and nonhayes. The default value is hayes.

NCSIExtended

A boolean parameter which, when set o on, enables you to set baud rate, parity, server name, general service name and specific service name. The default is off.

NCSIServer

Specifies the server name, using the following syntax:

```
set NCSIServer to "servername[*]"
```

NCSIGeneralService

Specifies the general service device name, using the following syntax:

```
set NCSIGeneralService to "devicename [*]"
```

NCSISpecificService

Specifies the specific service device name, using the following syntax:

```
set NCSISpecifiedService to "devicename [*]"
```

newFileName (prompt, default_file_name)

Displays a dialog box to allow the user to specify a file to create or overwrite, and returns the file name selected. If the user clicks Cancel, an empty string is returned.

```
MyFile# = newFileName (“File to create”,”myfile.txt”)
if myfile# <> “”
create file myfile#
endif
```

numberOfChars (expression)

Returns the number of characters in the given string expression.

```
testData# = first_second_third” + cr + lf + “line 2” + cr &
+ lf + “line 3, end”
display testData#
display “Data contains” numberOfChars (testData#) +&
“characters” + cr + lf
```

numberOfColumns

The number of columns in display memory. Possible values are 80, 132, or 160. The default value is 80.

numberOfItems (expression)

Returns the number of items in the given string expression.

```
testData# = first_second_third” + cr + lf + “line 2” + cr &
+ lf + “line 3, end”
display testData#
display “Data contains” numberOfItems (testData#) + “items” &
+ cr + lf
```

numberOfLines (expression)

Returns the number of lines in the given string expression.

```
testData# = first_second_third” + cr + lf + “line 2” + cr &
+ lf + “line 3, end”
display testData#
display “Data contains” numberOfLines (testData#) + “lines” &
+ cr + lf
```

numberOfScreens

The number of screens in display memory. The default value is 4. The maximum number is limited by the amount of available memory.

numberOfWords (expression)

Returns the number of words in the given string expression.

```
testData# = first_second_third” + cr + lf + “line 2” + cr &
+ lf + “line 3, end”
display testData#
display “Data contains” numberOfWords (testData#) + “words” &
+ cr + lf
```

numberToString (expression)

Converts the given numerical expression to a string depicting the decimal representation of the number. The contents of the string expression are evaluated until the first non_numeric character is encountered. The digits in the string are considered to be the decimal representation of the number. A null string or a string that does not begin with a digit returns a value of zero.

```
aNumber# = 10
aString# = numberToString (aNumber#)
```

numericPlusKeyResult

The current function of the + key. Possible values are returnKey, enterKey, and numericPlusKey.

ObjectProperties ( )

Returns a string supplying information about the properties of the current object. These properties are formatted one per line and can be accessed individually by using chunking expressions. The properties returned are, in order, name, type, creator, last writer, creation date, modify date, and comments. If no current object exists, returns error 500, noCurrentObject.

```
If line 2 of objectProperties() = "Text Note"
send object to "tempfile"
else
message prompt line 1 of objectProperties() &
+ " is not a Text Note" icon warn buttons "OK"
endif
```

open

Opens an application, or opens a file using the specified application.
Open application [with file] [maximized|minimized]
In a multi-application environment, the application is opened concurrent with MS92. In a single-tasking environment, MS92 terminates and the application is opened. When the application quits, MS92 is restarted but the script does not continue execution.

```
open "PageMaker"
open "Micorsoft Word 4.0" with "MyReport"
```

Errors

390 unableToOpenApplication
201 fileNotFound
205 fileBusy
100 insufficientMemory

open file

Opens a ASCII text file on the PC for reading and writing.
Open file file_name
A fully-qualified file name can be specified, with or without a file extension. If not otherwise specified, MS92 looks for the file in its own directory, unless this default had been overridden

by setting the directory configuration parameter or by changing directory in any file selection dialog box.
Two file pointers are maintained for the file: one for reading and one for writing. Initially, the read file pointer is positioned at the beginning of the file. Writing always takes place at the end of the file.

```
;open a user-specified file for read/write:
input into filename# = ""
stop
endif
open file filename#
```

Errors

200 filePermissionError
201 fileNotFound
204 tooManyFilesOpen
205 fileBusy
209 fileAlreadyOpen

Related Settings

```
directory
```

openScript

The name of the script automatically performed when the settings file is opened.

Open session

Opens a settings file and reads any text and graphics saved in the file into display memory.

```
open session file_name
open session "LAN900.SCF
```

The preceding command opens a window like this:
[pg.114]
Related Settings
openScript
Errors

200 filePermissionError
201 fileNotFound
202 fileIOError
204 tooManyOpenFiles
205 fileBusy
100 insuffucientMemory
361 badConfigFile

pageMode

A boolean parameter which, when set on, enables page mode. The default value is off.

page setup

Displays a dialog box in which the user can specify page setup options.
Page setup

paste

Pastes the contents of the clipbard into display memory at the current cursor location, or into the specified varaible.
Paste [into|after variable]
To paste at the current cursor location, specify paste with no additional parameters.
When pasting to a variable, specifying into replaces any data currently contained in that variable. Specifying after appends the data to the end of the current contents of the variable.

```
;Copy line 1 and paste its contents to line 10
select line 1
copy
cursor 10
paste
;Copy line 1 and paste it into the variable mylines#
select line 1
copy
paste into mylines#
;Select another line and append it to the variable
```

```
select line 10
copy
paste after mylines#
;Now mylines# contains lines 1 and 10
```

Errors

391 errorAccessingClipboard

phoneNumber

The default phone number to be dialed by the modem.

PhoneType

The type of dialing used when dialing the modem. Possible values are tone and pulse. The default value is tone.

Port

The serial port to be used. Possible values are com1, com2, com3, com4, port1, port2, port3, and port4. The default value is com1. Possible values are modem and printer, the default setting.

print

Prints a text file, the graphics screen, or the current log to the printer currently specified in the Macintosh Chooser.

```
Print file file_name
print graph
print log
;To print the contents of the file “myfile.txt”
print file “myfile.txt”
;Display a dialog box from which th
e user can print a file, the  ; contents of the graphics window,
or the current log.
message into printChoice#prompt “What would you like to print” &
buttons “FILE”, “LOG”, “GRAPH”
if printChoice# =1
```

```
;Use getFilename function to display a standard file open box.
;*.* means all files will be displayed.
print file getfilename ("File to print:","*.*")
elseif printChoice# =2
print log
else
print graph
endif
```

Related Settings

Directiory (for print file only)
draftPrintSize (for print log only)

Errors

350 printError
200 filePermissionError
201 fileNotFound
202 fileIOError
204 tooManyOpenFiles
205 fileBusy
100 insufficientMemory

quit

Terminates execution of MS92, as if Quit or Exit were selected from the File menu.

```
close
exit
quit
```

Related Settings

closeScript

read file

Reads from the current read file pointer of a disk file previously opened with the open file command or the create file command.

```
read file file_name [read_condition] into variable
```

where read_condition is one of the following:

```
until string
for number_of_units chunking_unit
```

where chunking_units is one of the following:

```
characters      character
char            chars
line            lines
```

The file name specified must exactly match the file name that was specified in the open file command or the create file command.

Reading continues until the read_condition is met. If no condition is specified, one line is read by default.

The string following the until keyword specifies a string of terminating characters. Reading stops when these characters are read. If the string is the empty string (“”), the command reads until the end of the file.

The expression following the for keyword specifies how many characters or lines to read.

Reading from a file where the read pointer is positioned at the end of file returns an empty string.

```
; Read a user-specified file into display memory
fileName# = getFileName (“File to display:”,”*.*”)
if fileName#  = “”
stop
endif
open file fileName#
read file fileName# for 1 line into currentLine#
while currentLine# <> “”
display currentLine# + lf + cr
read file fileName# for 1 line into currentLine#
endwhile
close file fileName#
;Read the data file one field at a time.
;The fields are comma seperated
;Display the fields one per line.
open file “MYDATA”
read file “MYDATA” until “,” into currentField#
while currentfield# <> “”
display currentField# + cr + lf
read file “MYDATA” until “,” into currentField#
endwhile
close file “MYDATA”
```

Related Settings

directory

Errors

200 filePermissionError
201 fileNotFound
202 fileIOError
205 fileBusy
208 fileNotOpen

receive

Transfers a file from the host (the remote file) to the workstation (the local file).

receive [protocol link3000 | link9000 | xmodem]

For transfer from an HP 3000:

receive remote_file [[to local_file] [as [text | wordwrap | binary | backup | restore]] [delete] [protocol link3000]

For transfer from an HP 9000:

receive remote_file [[to local_file] [as [text | binary | backup | restore |]] [delete] [protocol link9000]

For xmodem transfer:

receive remote_file [[to local_file] [delete] [protocol xmodem]

If no parameters are specified, receive operates as if it had been selected from the File menu, displaying a dialog box in which the user can enter a file name and specify conversation options. The transfer is performed using the protocol specified by the transferProtocol configuration setting, which defaults to link3000.

If only the protocol parameter is specified, the command operates interactively using the specified protocol.

If the local file name is omitted, the local file takes the same name as the remote file. If a local file with the given already exists, a runtime error occurs unless delete is specified to overwrite the existing file.

File conversion options are available for transfers from the HP 3000 or HP 9000. If the as parameter is omitted, the default conversion option depends on the type of file transferred. When used for HP 3000 transfers, the text option converts an HP 3000 EDIT / 3000 file into standard text format. All trailing blanks are stripped from each record, and a single carriage

return is appended. Non-printing characters, except tab, are replaced by a space. The default file extension .TXT is automatically added to the file name.

When used for HP 9000 transfers, the text option converts a text file, produced by many HP - UX editors, into a standard text file. In particular, it changes a newline, linefeed, or <LF> character, used to separate lines and paragraphs by HP - UX editors, into a carriage return <CR> and linefeed <LF> characters, used to separate lines and paragraphs by PC applications. The default file extension .TXT is automatically added to the file name for PC file transfer.

The wordwrap option, available for HP 3000 transfers only, converts an HP 3000 EDIT / 3000 file into standard text format. For more on wordwrap, see the Technical Reference.

The binary option copies any type of HP 3000 or HP 9000 file into a PC file. All data characters in each record of the HP 3000 file are written to the data file with no modifications. The default file extension .TXT is automatically added to the file name.

The backup option copies any type of HP 3000 or HP 9000 file onto a PC so that it can later be restored with the restore option of the send command. All data and user labels are copied, so that all file characteristics are preserved when the file is restored to the HP system. The file extension .HP3 or .HP9 is automatically added to the file name.

The restore option restores a PC file which has previously been archived on the HP 3000 or HP 9000 with the backup option of the send command. Data and file attributes are all restored.

Additional details on file transfer are found in the Technical Reference.

```
Receive “myfile”
;receives myfile using default protocol
;placing it on PC with same name
receive “myfile” to “newfile” delete
;receives myfile using default protocol, placing it on PC as
;newfile, deleting an old copy of newfile if it already exists
receive “myfile” as binary
;receives myfile using default protocolusi ng the binary
;conversion option
receive “myfile” protocol link9000
;receives myfile using 9000 protocol
```

Related Settings

baud
blockSize
compressXfer
dc1Pacing

defaultBinaryXfer
hostFileStartup
recode8BitXfer
recodeCtrlXfer
transferProtocol
transferTimeIncrement

Errors

100 insufficientMemory
105 stringTooLong
200 filePermissionError
202 fileIOError
203 fileExists
204 tooManyOpenFiles
205 fileBusy
206 fileTooLarge
207 diskFull
300 unableToRunLinkProgram
303 fileTransferUnsuccessful
304 fileTransferCanceled
305 hostfilePermissionError
306 fileNameTooLong
307 invalidLinkVersion

recode8bitXfer

A boolean parameter which, when set on, enables the recoding of 8-bit characters during file transfer. The default value is off.

recodeCtrlXfer

A boolean parameter which, when set on, enables the recording of control characters during file transfer. The default value is off.

redial

A boolean parameter which, when set on, enables automatic redialing until the modem until connected. The default value is off.

remote

A boolean parameter which, when set on, enables remote mode. The default value is off.

rename file

Renames a file on disk

```
rename file old_file_name to new_file_name
```

old_file_name must evaluate to the name of an existing disk file.
new_file_name must evaluate to a file name valid for the operating system. If a file with this name exists, an error occurs.

```
;Rename a user-specified file:
oldName# = getFileName ("File to rename:","*.*")
if oldName# = "" or not exist (oldName#)
stop
else
newName# = newFileName ( "Newfile name:","")
if newName# = ""
stop
else
rename file oldName# to newName#
endif
endif
```

Related Settings

directory

Errors

200 filePermissionError
201 fileNotFound
203 fileExists
205 fileBusy

reset

Resets the terminal

soft reset
hard reset

soft reset emulates the function of the reset key on an HP terminal. MS92's soft reset does the following:

- Display functions and Keyboard Lock are turned off.
- Any data communication transfers in progress are canceled.
- The bell rings.
- The screen is refreshed.

Hard reset has the same function as pressing Ctrl + Shift + Reset on an HP terminal. MS92's hard reset does the following:

- Display memory is cleared.
- All tabs are cleared. (Margins are not cleared.)
- The following configuration values and mode settings are reset to their default values:
  - Page mode off
  - Inhibit Line Wrap (strap C) off
  - Inhibit Handshake (strap G) off
  - inhibit DC2 (strap H) off
  - Transmit Functions off
  - Keyboard Lock off
  - Memory Lock off
  - Insert Character mode off
  - Display Functions off
- The typeahead buffer is cleared.
- Any datacomm transfers in progress are canceled and the buffers are emptied.
- Display Functions and Keyboard Lock are turned off.
- If enabled, record mode and logging are disabled.
- The bell rings.
- The screen is refreshed.

Related Settings emulation

restore

Restores MS92's windows to their normal size.

Restore [text|script]

If the MS92 application is iconized, the application is restored – all open windows are visible and the window specified in the restore command is brought to the front. The default it text.

Return

Terminates execution of current procedure or current script. If the current script was initiated by another script, returns control to initiating script.

```
return
proc printProcedure
fileName# = getFileName ("File:","*.*")
if fileName# = "" or not exist (fileName#)
return
endif
print file fileName#
endproc
do printProcedure
;when return in the above procedure is encountered, or when the
;procedurecompletes, script execution resumes at the following
;line
display "Complete"
```

returnKeyResult

The current function of the Return key. Possible values are returnKey and enterKey.

Revert

Reads the current settings file, returns all settings to the values specified therein, clears display memory, and performs a hard reset.

Revert

If no settings file is open, MS92's default configuration settings are restored.

Errors

100 insufficientMemory
200 filePermissionError
201 fileNotFound
202 fileIOError
204 tooManyOpenFiles
205 fileBusy
361 badConfigFile

rightMargin

The column position of the right margin. Must be a number between the value of leftMargin and the value of the numberOfColumns parameter. the default is the value of the numberOfColumns set parameter.

save

Saves MS92 data and / or settings to a file.

save text file_name [delete]
save graph file_name [as paint|draw] [delete]
save log file_name [delete]
save session file_name [delete]

save text saves the current contents of display memory to a disk file. If you add the extension .TXT to the file name you specify, the file can be opened by most Windows text processors.
Save graph saves the current contents of graphics memory to a disk file. If paint or draw is specified, the graphics are saved as a painting or drawing file respectively. Default is paint.
Save log saves the current log to a disk file.
save session saves the current configuration settings (all set command parameters and all options chosen using commands from MS92's Settings menu) to a disk file.
filename is the full path filename.

Drive_name:folder_name:file_name

In all forms of the command, the delete keyword can be specified to overwrite an existing file with the name specified. With save log, if the file exists and delete is not specified, the newly logged data is appended to the end of the file. With the other versions of the command, if the file exists and delete is not specified, a runtime error occurs.

```
;Save the contents of display memory to a file:
save FileName# = newFileName ("New File:","")
if saveFileName# <> ""
save text saveFileName#
endif
;Change some settings and save the session to a file:
set baud to 1200
set numberfColumns to 132
save session "MySessionName" delete
```

Related Settings

directory

Errors

100	insufficientMemory
200	filePermissionError
201	fileNotFound
202	fileIOError
203	fileExists
204	tooManyOpenFiles
205	fileBusy
206	fileTooLarge
207	diskFull

select

Selects or deselects parts of display memory.

```
select select_option
```

where select_option is one of the following:

```
upper_row, left_col, lower_row, right_col
line [number [to number]]
screen
all
cancel
```

A rectangular region of the screen can be specified by specifying four screen coordinates. The coordinates must be given in the order shown above. For example, select 1,2,23,80 would select the entire screen except the last line.

A single line or a range of lines can be specified with the line option. If line is specified alone, the line where the cursor is currently positioned is selected.
Select screen selects the visible portion of display memory.
select all selects the entire contents of display memory.
select cancel deselects any current selection.

```
;Select some lines in display memory and copy them to the
;clipboard
select line 1 to 10
copy
```

send

Transfers a file from the personal computer (the local file) to the host (the remote file).

```
send [protocol link3000 | link9000  | xmodem]
```

For transfer to an HP 3000:

```
send local_file [to remote_file] [as [text | wordwrap | binary
| backup | restore |]] [delete] [protocol
link3000] [record number bytes|words]
```

For transfer to an HP 9000:

```
send local_file [[to remote_file [as [text | binary | backup |
restore |]] [delete] [protocol link9000]
```

for xmodem transfer:

```
send local_file [[to remote_file] [delete] [protocol xmodem]
```

If no parameters are specified, send operates as if it had been selected from the File menu, displaying a dialog box in which the user can enter a file name and select conversion options. The protocol used is that specified by the transferProtocol configuration setting, which defaults to link3000.
If only the protocol parameter is specified, the command operates interactively using the specified protocol.
If remote_file is omitted, the remote file name is built from the local file name, using the first eight characters that are valid for a name on the host operating system. If a remote file with the given name already exists a runtime error occurs, unless delete is specified to overwrite the existing file.
For HP 3000 transfers only, record specifies the size for the remote file in either bytes or words.

File conversion options are available for transfers to the HP 3000 or HP 9000. IF the as parameter is omitted, the default conversion option depend on the type of file transferred.
When used for HP 3000 transfers, the text option converts a text file into standard HP EDIT/3000 format. By default, the records are 72 bytes, ASCII. Each line in the text file (terminated by a carriage return) is written to a single record in the HP 3000 file. If a line is too long to fit in the established record length, as many characters are placed in the record as will fit. Additional records are written until all characters have been copied. Carriage returns are stripped from the PC file. When consecutive carriage returns are encountered, each one after the first causes a blank record to be written to the EDIT/3000 file. All other non-printing characters including new-line, new-page, and so on, are replaced by a space. Tabs, however, are not replaced with a space.
When used for HP 9000 transfers, the text option converts a text file into a file suitable for use with HP-UX editors such as vi. In particular, it converts a text file which uses the carriage return <CR> and linefeed <LF> characters to separate lines or paragraphs into a file which uses the linefeed <LF> character to separate lines or paragraphs. This is the default option for text files.
The wordwrap option is available for transfers to the HP 3000 only. For more information on wordwrap, see the Technical Reference.
The binary option copies any type of file into an HP 3000 or HP 9000 binary file. Only the data is copied; the file attributes are not. The data in the file is sent to the host with no modifications. On the HP 3000, the default record length is 128 words, binary, and each record is completely filled with data. For example, if the HP 3000 record length is 128 words, the first 256 characters (bytes) in the PC file are written to the first record; the next 256 characters are written to the second record, and so on.
The backup option copies any type of file onto the HP 3000 or HP 9000 so that it can later be restored with the restore option of the receive command. Data and file attributes are copied, so that all characters are preserved when the file is restored.
The restore option restores an HP 3000 or HP 9000 file previously archived on the PC with the backup option of the receive command.
Additional details on file transfer are found in the Technical Reference.

```
Send "myfile";;sends pc-based myfile to host
                ;using default protocol, and
                ;places it in a file with the
                ;same name
```

```
send “myfile” to “newfile” delete
                        ;sends myfile using default
                        ;protocol placing it on host as
                        ;newfile, deleting old copy of
                        ;newfile if it already exists
send “myfile” as binary
                        ;sends myfile using default
                        ;protocol using the biniary
                        ;conversion option
send “myfile” protocol link9000
                        ;sends myfile using link9000
                        ;protocol
send “myfile” record 80 bytes ;sends myfile and places it on
                        ;host in a file called “myfile”
                        ;with 80 byte records
```

Related Settings

baud
blockSize
compressXfer
dc1Pacing
defaultBinaryXfer
hostFileStartup
recode8BitXfer
recodeCtrlXfer
transferProtocol
transferTimeIncrement

Errors

100 insufficientMemory
105 stringTooLong
200 filePermissionError
201 fileNotFound
202 fileIOError
203 fileExists
204 tooManyOpenFiles
205 fileBusy
300 unableToRunLinkProgram
301 illegalConversionOption
302 illegalRecordSpecification

303 fileTransferUnsuccessful
304 fileTransferCancelled
305 filePermissionError
306 fileNameTooLong
309 hostFileExists
310 invalidRecordSize
307 invalidLinkVersion

sendline

Sends a string to the host followed by a carriage return.

Sendline string

To receive a response from the host, use the expect command immediately following sendline. If multiple lines of data are sent, and if the waitHostPrompt configuration parameter is on, the second and subsequent sendline are sendstring commands in a script will wait for prompt from the host before sending data. If waitHostPrompt is off, neither command waits for the host prompt. The host prompt is set with the hostPrompt configuration parameter.

```
The maximum length of the string sent is 32,767 characters.
;Sending data to the host:
input into logonString# prompt “Enter you logon command:”
input password into passwordString# prompt&
“Enter you password:”
set hostPrompt to dc1
sendline logonString#
expect dc1 wait 10 seconds
if passwordString# <> “”
sendline passwordString#
expect dc1
endif
input into command# prompt “Enter command to be performed:”
if command# <> “”
sendline command#
endif
sendline “bye”
```

Related Settings

autoLineFeed
escDelayAmount

hostprompt
localEcho
typeAhead
waitHostPrompt

Errors

105 stringTooLong

sendstring

Sends the specified string to the host computer without a carriage return

sendstring string

To receive a response from the host, use the expect command immediately following sendstring. If multiple lines of data are sent, and if the waitHostprompt configuration parameter is on, the second and subsequent sendline or sendstring commands in a script wait for a prompt from the host before sending data. If waitHostPrompt is off, neither command waits for the host prompt. The host prompt is set with the hostPrompt configuration parameter.

The maximum length of the string sent is 32,767 characters.

```
;This command can be used to build a command from separate parts
sendstring 'LISTF"
sendstring ",2"
sendline ""
```

Related Settings

autoLineFeed
escDelayAmount
hostpromt
localEcho
typeAhead
waitHostPrompt

Errors

105 stringTooLong

sessionName

The symbolic name of the current session. By default, it is the name of the open settings file.

set

Changes a setting to the value specified. Applies to the focus session (the session from which the current script was run) only.

Set [the] setting to expression

Settings are listed on the next page.

set [the] [screen] row to row_number
set [the] [screen] column to column_number

where row_number indicates the target row, and can be any number from one to the limit of your display memory.
where column_number indicates the target column, and can be any number from one to 60 (depending on you display setting)

set [the] window_type location to location_string
set [the] window_type rectangle to rectangle_string
set [the] window_type title to title_string

where window_type is one of the following:

text | graphics | script

where location_string provides the coordinates of the upper left corner of the window in the format:

"upper_pixel, left_pixel"

where rectangle_string provides the coordinates of the upper left and lower right corners of the window in this format:

"upper_pixel, left_pixel, lower_pixel, right_pixel"

set [the] fkey fkey_descriptor to expression
where fkey is one of the following:

f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12

where fkey_descriptor is one of the following:

label | message | script | attribute

where setting is one of the following:

alternateSet	autoHorizscroll	autoKeyboardLock	
autoLineFeed	backspaceKeyResult	baud	
blockMode	blocksize	characterDelay	
closeScript	compressXfer	connection	
ctrlAltIsExtChar	cursorType	dataParityBits	
dc1Pacing		defaultBinaryXfer	
directory	disableBell	disableEnh	

disableHostControl display displayFunctions
draftPrintSize emulation enqAck
enterKeyResult escDelayAmount fontSize
formatMode formsCache graphResolution
graphScaling graphWindows hardHangup
helpFile hostFileStartup hostName
hostPrompt inhibitDc2 inhibitHandshake
inhibitLineWrap keyboardLock language
leftMargin lineModify localEcho
lockFkeys logDirection localEcho
lockFkeys logdirection maximizeSiz
memoryLock modemType NCSI
NCSIExtended NCSIGeneralService NCSIserver
NCSISpecific-Service
numberOfColumns numberOfScreens
numericPlusKey-result
objectScript openScript
pageMode phoneNumber phoneType
port recode8bitXfer recodeCtrlXfer
redial remote returnKeyResult
rightMargin sessionName shiftBackspace-Result
smoothScroll spacesPerTab tabKeyResult
terminalId timeout transferProtocal
transferTimeIncrement
typeAhead waitHostPrompt
xcmodemBinary xmitFunctions xmodem1Kbinary
xmodem1KCRC16 xmodem1KGBinary xmodemCRC16
xonXoffInput xonXoffOutput ymodem1KBlocks
ymodemGBinary ymodsmBinary zmodemCRC32
zmodemEncodeCtrls zmodemFileMgmnt zmodemFullPath
zmodemResume zmodemTimeout zmodemUpdateOnly
zmodemWindowSize

The setting parameters allow you to set various configuration items. For a complete description, see each setting's individual listing.
The window parameters allow you to change the current location, rectangle, and title of each window.
The row and column parameters allow you to set the memory-relative position of the cursor. To set the position of the cursor relative to the portion of memory currently displayed on the screen, add the screen keyword.
set fkey allows you to change the current values of the label, message, script, or attribute associated with a specific function key. Valid attributes include normalAttribute, localAttribute, transmitAttribute, or scriptAttribute.

```
;Save current config setting, and then set new ones
proc saveSettings
get the baud into oldBaud#
endproc
proc changeSettings
set the baud to stringToNumber (newBaud#)
set the numberOfScreens to
stringToNumber (newNumberOfScreens#)
endproc
do saveSettings
input into newBaud# prompt "Enter value for new baud rate:" with &
numberToString (oldBaud)
input into newNumberOfScreens# prompt "Enter new number of&
screens: with numberToString (oldNumberOfScreens)
do changeSettings
;Using set to size text window
input into response# prompt &
"Enter rectangle coordinates: four numbers, separated by commas:"
upperRow# = stringToNumber (item 1 of response#)
leftCol# = stringToNumber (item 2 of response#)
lowerRow# = stringToNumber (item 3 of response#)
rightCol# = stringToNumber (item 4 of response#)
set the text rectangle to&
upperRow#, leftCol#, lowerRow#, rightCol#
input into newTitle# prompt "Enter new text window title:"
if numberOfChars (newTitle#) <> 0
set the text title to newTitle#
endif
;using set to attach a script to a function key, and provide
```

```
;appropriate label
set f1 script to "logon.TTS"
set f1 label to "LOGON"
```

Note that you can use permanent variables, as described on page 22, so that the value of that variable is not reset each time a script is run.

Errors

102 invalidSetValue
103 setCommandFailed
105 stringTooLong

setup

Displays a dialog box in which the user can change MS92's configuration.

Setup function [result vatiable]

where function is one of the following:

```
color
connection
fkeys
graphics
keyboard logging
scripts
terminal
transfer
```

To monitor whether the user selected the OK or Cancel button on the dialog box, specify the result keyword. If OK is selected, a value of 1 is placed in the variable. Otherwise, it contains a value of 0.

```
;Display dialogs to allow user to change configuration
;settings
If any changes are made, save settings in a settings file.
setup terminal result response1
setup transfer result response2
setup keyboard result response3
If response1 =1 or response2=1 or response3=1
Save session "newconfig"
endif
endif
```

The preceding script displays the Terminal Settings, File transfer Settings, and Keyboard Settings dialog boxes. See the MS92 Technical Reference for complete details on these dialog boxes.

ShiftBackspaceResult

The current function of the Shift-Backspace key combination. Possible values are backspaceKey, which performs a backspace without deleting the character to the left of the cursor, destructiveBackspace, which performs a backspace followed by a space then another backspace, deleting the character to the left of the cursor, or delCharacter, which is utilized by some applications.

Show

Makes the specified display element visible to the user.

Show controlBar
show buttons
show graphics
show controls
show text
show tabs
show fkeys
show script
show indicators

Objects are hidden with the hide command. If the specified object is already vivsible, show command does nothing.
Show controlbar displays the buttons under the menu bar.
[pg.139]
show buttons, show controls, and show tabs can be used to rotate the control bar display, showing edit buttons, control character buttons, or the margin/tab ruler. Show graphics works with Graphic MS92 for Macintosh. In two-window mode, it opens the graphics window, or it brings it to the front. In one-window mode, it displays the graphics image.
In two-window mode, show text opens the text window, orit brings it to the front. In one-window mode, it displays the text.
Show fkeys displays the on-screen function keys.

[pg, 140 (1)]
show script open the script window.
Show indicators displays the row and column counters, connection, keyboard lock, logging, and script indicators at the lower left of the window.
[pg, 140 (2)]
Related Settings
lockFkeys (For show fkeys only.)

smoothScroll

A boolean parameter which, when set on, enables smooth scrolling. The default value is off.

spacesPerTab

The number of spaces produced by the Tab key when it is tabKeyResult is set to spaces. See tabKeyResult.

Stop

Terminates execution of all currently-active scripts
stop

```
;Stop execution if the user does not input a file name in
;response to prompt input into filename#
prompt "enter desired file name:"
if filename# = ""
stop
endif
open file filename#
```

stringToNumber (expression)

Converts the given string to a numerical value. The contents of the string expression are evaluated until the first non-numeric character is encountered. The digits in the string are

considered to be the decimal representation of a number. A null string or a string that does not begin with a digit will return a value of zero.

```
aString# = “123”
aNumber# = 123
if stringToNumber (aString) = aNumber#
display “Equal!” + cr + lf
endif
```

tab

Sets a tab at the specified column.

```
Tab [at column_number]
```

If no column is given, a tab is set at the current cursor column.

```
;Set a special group of tab settings:
MyTabSettings# = “7,8,12,16,20,24,28,32,36”
I = 1
while I <= numberOfItems(MyTabSettings#)
thisTab# = stringToNumber (item I of MyTabSettings#)
tab at thisTab#
I = I + 1
endwhile
```

tabKeyResult

The current function of the Tab key. Possible values are tabKey, returnKey, and spaces.

Terminalid

The HP terminal ID string, usually 70094, 2392A, 2624A, 2622A.

Time ( )

Returns the current time in HH:MM:SS format, based on a 24-hour clock.

```
Display “The time is “ + time () + cr + lf
```

timeout

The default timeout in seconds for the expect command. The default value is 60 seconds.

TransferProtocol

The protocol to be used for file transfers. Possible values are link3000, link9000, xmodem, xmodem1K, xmodem1KG, ymodem, ymodemG, and zmodem. The default value is link3000.

TransferTimeIncrement

The amount of time in milliseconds by which MS92 increases its time delay for every packet not successfully received during an HP 3000 file transfer in which DC1 support is disabled (normally, over X.25). The default value for transferTimeIncrement is 100 ms. MS92 begins non-DC1 file transfers using a 200 millisecond delay before sending a packet to the host. If a packet is not successfully received, the transferTimeIncrement is added to the delay. If successive packets are not successfully received, the transferTimeIncrement is again added to the delay until a packet is successfully received. In this way MS92 adapts to the response time of the host system.

For networks or hosts with very slow response the transferTimeIncrement may need to be set to 500 or greater. This means that MS92 will wait an extra 500 ms every time a packet is not successfully received by the host. For example, the delay starts at 200 ms. If the first packet is not received, MS92 adds 500 ms to the delay, causing a total delay of 700 ms. If that time is still too fast and the packet is again unsuccessfully received, another 500 ms is added, making a total delay of 1200 ms. This process continues to use the delay which produced a successful result for all file transfers until MS92 is excited.

Normally, the default value is appropriate for most systems. The value of transferTimeIncrement should be kept as low as possible, since the larger the value, the slower file transfers will run.

Trap errors

Reinstates error trapping after ignore errors has been specified, so that whenever an error is found, a message is displayed and the script terminates.

Trap errors

If a non-fatal error occurs, MS92's default is to report the error and terminate execution of all currently active scripts. However, you can override this default by specifying the ignore errors command. To return to default operation, specify trap errors. If scripts have been nested, ignore errors will remain in effect until the highest level script terminates, or until a trap errors statement is encountered.

```
Ignore errors                    ;do not quit if an error occurs
open file "MYFILE"
if error() : If error occurs on open file
create file "MYFILE"             ;file doesn't exist. Try to
                                 ;create it.
If error() = 203            ;Check for the error
                                 ;"FileExists"
display "That file already exists."
Stop
endif
endif
trap errors                      ;Monitor errors again
```

upper (expression)

Converts all lower case characters in the source string to upper case and returns the resulting string. All characters not belonging to the set A-Z are unaffected. The source string is not altered by this function.

```
allower# = "abcdefg"
allUpper# = "ABCDEFG"
lowerResult# = lower (allUpper#)
upperResult# = upper (allLower#)
if allLower# = lowerResult# and allUpper# = upperResult#
display "Successful!" + cr + lf
endif
```

wait

Pauses script execution until a specified time.

```
Wait until time_string
```

where the format of time_string is "HH:MM:SS"

wait number_of_units time_unit

where time_unit is one of the following:

second[s] | minute[s] | hour[s]

The time can be specified as a time of day (use 24-hour military clock), or as a number of seconds, minutes, or hours. If no timespec is given, the number of seconds specified in the timeout configuration setting is used.

```
wait until “12:01”
do script “backup”
;here, the time is stored in a variable
backuptime# = “01:30”
wait until backuptime#
do script “backup”
;or a delay can be specified
sendline “hello manager.sys”
wait 10 seconds
sendline “password”
```

waitHostPrompt

A boolean parameter which, when set on causes the sendline and sendstring commands to wait for the prompt specified by the hostPrompt parameter before sending data. The default value is on.

xmitFunctions

A boolean parameter which, when set on, enables the transmit functions (A) strap. The default value is off.

xonXoffInput

A boolean parameter which, when set on, enables XON/XOFF handshaking on input from serial port. The default value is off.

xonXoffOutput

A boolean parameter which, when set on, enables XON/XOFF handshaking on output to serial port. The default is off.

Appendix A
Control Characters and Constants

Control characters

In general, to send a control character from a TermTalk script, you use the standard alphanumeric control character abbreviations shown in the first column of the table below. However, when specifying a control character within a quoted string, you must take a different approach. Because the abbreviation is transmitted literally rather than acted upon if placed within quotes, a caret plus the appropriate letter or symbol should be specified instead.

outside quotes	**inside quotes**	**decimal value**
NUL	^@	0
SOH	^A	1
STX	^B	2
ETX	^C	3

EOT	^D	4
ENQ	^E	5
ACK	^F	6
BEL	^G	7
BS	^H	8
HT	^I	9
LF	^J	10
VT	^K	11
FF	^L	12
CR	^M	13
SO	^N	14

outside quotes	**inside quotes**	**decimal value**
SI	^O	15
DLE	^P	16
DC1	^Q	17
DC2	^R	18
DC3	^S	19
DC4	^T	20
NAK	^U	21
SYN	^V	22
ETB	^W	23
CAN	^X	24
EM	^Y	25
SUB	^Z	26
ESC	^[	27
FS	^\	28

GS	^]	29
RS	^^	30
US	^-	31

For example, here are two ways to specify the word hello followed by a carriage return and a line feed.
“hello” + cr + lf
or
“hello^M^J”
When specifying a control characters inside or outside quotes, case is not significant. Upper or lower case letters can be used.

Constants

TermTalk defines the following terms as constants. You should ensure that none of your variable names or procedure name conflict with these constants. If a variable does have the same name as a TermTalk constant, it must be terminated with a # symbol.

ACCEPT	ADD	ADVANCEMENT
ADVISE	AFTER	AGENT
AGENTTASKRUNNING	ALL	ALTERNATESET
AND	APPLICATION	AS
ASCII7	ASCII8	AT
ATTRIBUTE	AUTOHORIZSCROLL	AUTOKEYBOARDLOCK
AUTOLINEFEED	BACKSPACEKEY	BACKSPACEKEYRESULT
BACKTAB	BACKUP	BAPI
BAUD	BEEP	BINARY
BLACK	BLOCKCURSOR	BLOCKMODE

BLOCKSIZE	BOLD	BOTTOM
BREAK	BUSY	BUTTONS
BYTES	CANCEL	CHANGE
CHAR	CHARACTER	CHARACTERDELAY
CHARACTERS	CHARS	CHECK
CLEAR	CLIPBOARD	CLOSE
CLOSESCRIPT	COLOR	COLUMN
COM1	COM2	COM3
COM4	COMMAND	COMPILE
COMPRESSED	COMPRESSXFER	CONNECT
CONNECTED	CONNECTION	CONTAINS
CONTROLBAR	CONTROLS	COPY
COPYSCREEN	CREATE	CTRLALTISEXTCHAR
CURRENTOBJECT	CURSOR	CURSORTYPE
CUT	DANISH	DATA
DATAPARITYBITS	DATA	DC1PACING
DDE	DDEAPPLICATION	DDECONVERSATION
DDEDATA	DDEEXECUTE	DDEINITIATE
DDEITEM	DDEMASSAGE	DDEPOKE
DDEREQUEST	DDESERVER	DDETEMPLATE
DDETERMINATE	DDETIMEOUT	DEFAULTBINARYXFER
DELCHARACTER	DELETE	DESTRUCTIVEBACK-SPACE
DIAL	DIALOG	DIRECTORY
DIRNAME	DISABLE	DISABLEBELL
DISABLEENH	DISABLEHOST-CONTROL	DISCONNECT
DISPLAY	DISPLAYFUNCTIONS	DO

DOWN	DRAFTPRINTSIZE	DRAW
EICON	EIGHTNONE	ELSE
ELSEIF	EMULATION	ENDIF
ENDOFFILE	ENDPROC	ENDWHILE
ENHANCEMENTS	ENQACK	ENTER
ENTERKEY	ENTERKEYRESULT	ERROR
ERRORLINE	ERRORS	ERRORSTRING
ESCDELAYAMOUNT	EVEN	EXECUTE
EXIST	EXIT	EXPANDED
EXPECT	EXPORT	F1
F10	F11	F12
F2	F3	F4
F5	F6	F7
F8	F9	FALSE
FIELDS	FILE	FILES
FILESPECIFICATION	FIND	FINNISH
FIT	FKEYS	FONTSIZE
FOR	FORM	FORMATMODE
FORMSCACHE	FREEDISK	FREEMEM
FRENCH	FROM	FULLSCREEN
GERMAN	GET	GETFILENAME
GOTO	GRAPH	GRAPHICS
GRAPHRESOLUTION	GRAPHSCALING	GRAPHWINDOWS
HANDLERS	HARD	HARDHANGUP
HAYES	HELPFILE	HIDE
HIGH	HOMEDOWN	HOMEUP

HOSTFILEDSTATUP	HOSTNAME	HOSTPROMPT
HOUR	HOURS	HP264X
HPANSI	HPCOLORGRAPHICS	HPGRAPHICS
HPTEXT	ICON	IDENTIFY
IF	IGNORE	IMPORT
INDICATORS	INHIBITDC2	INHIBITHANDSHAKE
INHIBITLINEWRAP	INITIATE	INPUT
INSERT	INSTALLHP 3000	INSTALLHP 9000
INT14	INTERACTIVE	INTO
ISEMPTY	ISNUMBER	ISSTRING
ITALIC	ITEM	KEY
KEYBOARD	KEYBOARDLOCK	LABEL
LANGUAGE	LEFT	LEFTMARGIN
LENGTH	LINE	LINECURSOR
LINEDRAWING	LINEMODIFY	LINEMODIFYCOLUMN
LINES	LINK3000	LINK9000
LIST	LOCALATTRIBUTE	LOCALECHO
LOCATION	LOCK	LOCKFKEYS
LOG	LOGDIRECTION	LOGGING
LOW	LOWER	MACBINARY
MAKE	MARGINS	MATH
MAXIMIZE	MAXIMIZED	MAXIMIZESIZE
MEMORYLOCK	MENU	MESSAGE
MINIMIZE	MINIMIZED	MINUTE
MINUTES	MOD	MODEM
MODEMPORT	MODEMTYPE	NAME

NEW	NEWFILENAME	NEWWAVE
NEWWAVEPACKAGE	NEXTPAGE	NO
NONE	NONHAYES	NORMAL
NORMALATTRIBUTE	NORWEGIAN	NOT
NOTE	NUMBEROFCHARS	NUMBEROFCOLUMNS
NUMBEROFIEMS	NUMBEROFLINES	NUMBEROFSCREENS
NUMBEROFWORDS	NUMBERTOSTRING	NUMERICPLUSKEY
NUMERICPLUSKEY-RESULT	OBJECT	OBJECTEXISTS
OBJECTFILE	OBJECTPROPERTIES	OBJECTS
OBJECTSCRIPT	ODD	OF
OFF	ON	OPEN
OPENSCRIPT	OR	PACK
PAGE	PAGEBREAK	PAGEMODE
PAGESETUP	PAINT	PASSWORD
PASTE	PEN	PERMANENT
PF1	PF2	PF3
PF4	PF5	PF6
PF7	PF8	PHONENUMBER
PHONETYPE	POKE	PORT
PORT1	PORT2	PORT3
PORT4	PREVPAGE	PRINT
PRINTERPORT	PROC	PROMPT
PROPORTIONAL	PROTOCOL	PULSE
QUERY	QUIT	RBLINE
READ	RECEIVE	RECODEXBITXFER
RECODECTRLXFER	RECORD	RECTANGLE

REDIAL	REJECT	RELATIVE
REMOTE	REMOVE	RENAME
REQUEST	RESET	RESPOND
RESTORE	RESULT	RESUME
RETURN	RETURNKEY	RETURNKEYRESULT
REVERT	RIGHT	RIGHTMARGIN
ROLLDOWN	ROLLLEFT	ROLLRIGHT
ROLLUP	ROW	SAVE
SCALED	SCREEN	SCRIPT
SCRIPTATTRIBUTE	SCRIPTS	SCRIPTSPEED
SECOND	SECONDS	SELECT
SELECTION	SEND	SENDLINE
SENDSTRING	SERIAL	SESSION
SESSIONNAME	SET	SETUP
SEVENEVEN	SEVENODD	SEVENONES
SEVENZEROS	SHIFT	SHIFTBACKSPACERESULT
SHOW	SMOOTHSCROLL	SOFT
SPACES	SPACEBAR	SPANISH
START	STOP	STRINGTONUMBER
SWEDISH	TAB	TABKEY
TEBKEYRESULT	TABLE	TABS
TASK	TELNET	TERMINAL
TERMINALID	TERMINALSCREEN	TERMINATE
TEXT	THE	TIME
TIMEOUT	TIMES	TITLE
TO	TONE	TOP

TOPIC	TRANSFER	TRANSFERPROTOCOL
TRANSFERTIME-INCREMENT	TRANSMITATTRIBUTE	TRAP
TRUE	TYPE	TYPEAHEAD
UK	UNADVISE	UNTIL
UP	UPPER	VT100
WAIT	WAITHOSTPROMPT	WARN
WHEN	WHILE	WINDOW
WINDOWIMAGE	WITH	WORD
WORDS	WORDWRAP	WRAP
WRITE	XMITFUNCTIONS	XMODEM
XONXOFFINPUT	XONXOFFOUTPUT	YES

Appendix B
Error Messages

Non-fatal errors

The following is a list of non-fatal TermTalk errors. The error number is found in the left column, followed by the error name and the error message. Please see the section titled "Error Handling" in Chapter 2 fir information on how to trap and retrieve non-fatal errors.
If you trap and receive a fatal error, cell your support representative.

100 insufficientMemory
There is not enough memory to do this command.
Suggestion: If you are running several other applications, try quitting from one or more of the other applications before running the script.

101 compileUnsuccessful
The script file was not compiled successfully.
Suggestion: Correct the error in the script file that you are trying to compile.

102 invalidSetValue
An invalid value was given for this particular set parameter.
Suggestion: Check valid values for this configuration setting.

105 stringTooLong

A string supplied to the command was too long. Strings cannot be longer that n characters.
Suggestion: Change the string so that it is no greater than the number of characters specified in the error message. The maximum length varies depending on the use on he string.

200 filePermissionError

The file's permissions do not allow this type of access.
Suggestion: Make sure that the operation that you are performing on the file is valid. For example, you cannot copy to an existing file if it is a read only file.

201 fileNotFound

The specified file could not be found.
Suggestion: Make sure you have specified the correct directory and file name.

202 fileIOError

A file system IO error occurred.
Suggestion: Make sure the file(s) specified in the command are correct, their permissions are correct, and so on.

203 fileExist

The specified file exits.
Suggestion: If you are trying to create a new version of an existing file, delete the existing file first.

204 tooManyOpenFiles

There are too many open files.
Suggestion: Close one or more files and try again.

205 fileBusy

This file is in use by another application.
Suggestion: Close the file from the other application and try again.

206 fileTooLarge

This file is too large

207 diskFull

The disk is full.
Suggestion: Delete any unnecessary files from your disk and try again.

208 fileNotOpen

The specified file is not open. You must open the file first
Suggestion: Make sure you have opened the file before trying to read, write, or close it.

209 fileAlreadyOpen

The specified file is already open. You cannot open the same file more that once.

Suggestion: Make sure you are trying to open the correct file.

250 connectFailed

An error prevented the connection from being established.

Suggestion: Check your connection hardware and configuration parameters.

260 modemNotResponding

The modem is not responding.

Suggestion: Make sure the modem is connected and turned on. Also make sure all of the connection settings are correct.

270 expectTimeout

A timeout has occurred. The expected string was not received before specified time expired.

Suggestion: Make sure you are expecting the correct string. If so, increase the amount of time the expect command waits by using the set timeout command.

300 unableToRunLinkProgram

Unable to run the file transfer program on the host computer.

Suggestion: Select File Transfer from the Settings menu to verify the host startup sequence and transfer method. Make sure you are logged on to the host computer.

302 illegalConversionOption

The specified conversion option is illegal for this file type.

Suggestion: change the conversion option specified to a valid option, or omit the “as” parameter and use the default conversion.

303 illegalRecordSpecification

The record unit specified (bytes or words) is illegal for this type of file conversion.

Suggestion: Change the record unit to the correct option or omit this parameter to use the default.

304 fileTransferCancelled

The file transfer operation was canceled.

305 hostFilePermissionError

The permissions of the host file prevented the successful transfer of the file.

Suggestion: Make sure the host file has the correct permissions and try again.

306 fileNameTooLong

The specified file name is too long. He maximum file name length is 36 characters.
Suggestion: Make sure the host file name specified contains 36 characters or less.

307 invalidLinkVersion

The version of the link program on the host computer is invalid.
Suggestion: Make sure you are running the correct version of Tymlink. Make sure the host file startup sequence contains the group and name of the Tymlink program. Check the hostFileStatrup configuration setting.

308 hostFileNotFound

The host file could not be found.
Suggestion: Make sure you specified the correct file name and location for host file.

309 hostFileExists

The host file already exists.
Suggestion: Add the delete option to your send command.

310 invalidRecordSize

The specified record size is invalid.
Suggestion: You cannot change the record size for some types of file formats such as backup. Remove the record specification and try again.

350 printError

A printer error occurred.
Suggestion: Make sure the printer is turned on and is online.

360 invalidSession

The session that was specified is invalid.
Suggestion: Make sure the session file you specified still exists and has the name you specified.

361 badConfigFile

The configuration file format is invalid.
Suggestion: Make sure the file you are loading is a MS92 configuration file.

380 notAScriptFile

The file specified is not a valid compiled TermTalk script file.
Suggestion: Make sure the script file is actually a compiled TermTalk file.

390 unableToOpenApplication

The specified application could not be opened.
Suggestion: Make sure you specified the correct path and file name of an executable file.

391 errorAccessingClipboard

An error was encountered while attempting to manipulate the clipboard data.
Suggestion: The Clipboard has memory limitations. Try copying a smaller amount of data.

400 ddeRejected

The DDE transaction was rejected by the remote application.
Suggestion: Make sure the transaction that you send is a valid transaction for the remote application.

401 ddeBusy

The remote application was busy and could not accept the DDE transaction.
Suggestion: Wait until the remote application is not busy and try sending the transaction again.

402 ddeError

A DDE error was encountered.
Suggestion: Ensure that the application you are trying to converse with is running and is monitoring DDE data.

403 ddeNoServer

Unable to find a remote application satisfying the DDE INITIATE command
Suggestion: Make sure there is another Windows application running that supports DDE and that you have specified parameters for the dde initiate command which include that application.

404 ddeTimeout

A timeout occurred in a DDE command. The remote application failed to respond before the specified time expired.
Suggestion: The remote application may be busy but failed to respond with a busy message. Try sending the transaction again. If you want the command to wait longer, use the set ddeTimeout command.

405 ddeNoHandler

An attempt to set up a DDE handler script failed.
Suggestion: Make sure the script is a valid TermTalk script and try again.

406 ddeIllegalResponse

An invalid use of the dde respond command occurred.
Suggestion: The dde respond command accepts only ACCEPT, REJECT, BUSY, or a string expression. Make sure you did not specify a number.

407 badConversationNumber

Conversation number is incorrect.
Suggestion: use the value returned by ddeinitiate.

500 noCurrentObject

501 invalidDOSFilename
An invalid DOS file name was specified.
Suggestion: Make sure you specified a valid DOS path and file name.

502 invalidObjectName
An invalid Object Name was specified. The name may be too long.
Suggestion: Check that the length of the object name is 32 characters or less.

503 exportFailed

504 inportFailed
Unable to import a NewWave object from the DOS file.
Suggestion: Make sure the DOS file exists. Ensure that there is sufficient space on the disk and that a disk is in the disk drive.

505 renameFailed
Unable to rename the NewWave object.
Suggestion: Check that the length of the object name is 32 characters or less.

506 hostFileNotObject
The specified host file is not a NewWave SOF Package.
Suggestion: Make sure you specified the correct host file name. On an HP 3000, a SOF file is shown with a file code of SOF in a LISTF directory listing.

507 pcFileNotObject
The specified PC file is not a NewWave SOF Package.
Suggestion: Make sure you specified the correct PC file name. You should be able to use the Import Object command of the NewWave Office Objects menu if this file is in the correct format.

508 agentAlreadyRunning
Unable to start an agent task because another task is already running.
Suggestion: Wait until the agent task has completed and try again. Use the agentTaskRunning() function to test this within a script.

509 agentTaskNotFound

Unable to locate the specified Menu Task.
Suggestion: Make sure the task you specified is listed in the Manage Menu Task dialog box, chosen from the Task menu.

510 currentObjectNotTask

The current NewWave object is not an Agent Task.
Suggestion: The object pasted, dropped, imported or received was not an Agent Task. Use the objectProperties() function to check the type of object before attempting to run it as a task.

Compiler errors

The following is a list of TermTalk compiler errors. Most of these are self-explanatory. When such an error is encountered, the compiler terminates and the appropriate message is displayed in a dialog box. If you received an error prefixed with the label Internal Error Cnn:, call you support representative.

An error was encountered while loading the compiler's internal tables.
Suggestion: Try quitting from some other applications to free up memory and try again.

An error was encountered while opening the file named *filename*.

A Syntax Error was encountered.

A non-terminal string was encountered. A string must have a closing quote A non-terminal string was encountered. A string must have a closing quote A GOTO statement was used to branch to *xxxx* which is an unknown label.

An ENDIF statement was expected. Each IF statement must have a corresponding ENDIF statement.

An ENDWHILE statement was expected. Each WHILE statement must have a matching ENDWHILE statement.

An identifier must represent only one VARIABLE, PROCEDURE, or LABEL. *label* is used to represent more that one.

A LABEL name cannot be used in an expression.

A PROCEDURE name cannot be used in an expression.

A '+' cannot be applied to a string constant.

A '-' cannot be applied to a string constant.

Chunking only applies to strings. You used chunking on a number.
'*' cannot be used with strings.
'/' cannot be used with strings.
'MOD' cannot be used with strings.
You cannot add a string and a number together.
'-' cannot be used with strings.
A chunking range value must be a number.
'CONTAINS' cannot be used with numbers.
'=' Must be applied to operands of the same type.
'<>' Must be applied to operands of the same type.
'<' Must be applied to operands of the same type.
'<=' Must be applied to operands of the same type.
'>' Must be applied to operands of the same type.
'>=' Must be applied to operands of the same type.
'NOT' cannot be used with strings.
'AND' cannot be used with strings.
'OR' cannot be used with strings.
LABEL names are unique. The LABEL name *labelname* is not unique.
The name *labelname* used as a LABEL already been used to represent a VARIABLE or PROCEDURE.
An ENDIF statement is invalid here.
An ELSEIF statement is invalid here.
The expression in an ELSEIF statement must evaluate to a number.
An ELSE statement is invalid here.
The expression in a WHILE statement must evaluate to be a number.
An ENDWHILE statement is invalid here.
A PROCEDURE must be defined before it is called.
It is invalid to assign a value to a LABEL.
It is invalid to assign a value to a PROCEDURE.
The menu name must be a string.
The item name must be a string.
The menu ID must be a number.
A script name must be a string.

You cannot send a number.
Illegal use of a label.
Illegal use of a procedure.
The prompt must be a string.
The expression following the ‘WITH’ keyword must be a string.
The length expression must be a number.
A button name must be a string.
The name of the new item must be a string.
The check expression must be a number.
The disable expression must be a number.
A string was found where a number was expected.
A number was found where a string was expected.
The phone number must be a string.
The EXPECT expression must be a string.
The expression following the ‘WAIT UNTIL’ keywords must be a string.
The expression following the ‘WAIT’ keyword must be a number.
The expression being sent to the host must be a string.
The expression specifying the protocol must be a number.
The filename must be a string.
The object name must be a string.
The record size must be a number.
The session name must be a string.
The application name must be a string.
The topic name must be a string.
The conversion specifier must be a string.
The object type must be a string.
The first parameter of the ‘Find’ function must be a string.
The second parameter of the ‘Find’ function must be a string.
The third parameter of the ‘Find’ function must be a number.
The ‘Lower’ function cannot operate on a number.
The ‘StringToNumber’ function cannot operate on a number.
The ‘NumberOfChars’ function cannot operate on a number.
The ‘NumberOfItems’ function cannot operate on a number.

The 'NumberOfLines' function cannot operate on a number.
The 'NumberOfWords' function cannot operate on a number.
The 'NumberToString' function cannot operate on a string.
The 'Upper' function cannot operate on a number.
The 'Exist' function cannot operate on a number.
The first parameter of the 'GetFileName' function must be a string.
The second parameter of the 'GetFileName' function must be a string.
The first parameter of the 'NewFileName' function must be a string.
The second parameter of the 'NewFileName' function must be a string.
The parameter of the 'ErrorString' Function must be a number.
The first parameter of the 'ObjectExists' function must be a string.
The second parameter of the 'ObjectExists' function must be a string.
The compiler was aborted by the user.

Index

D

E

F

G

O

P

Q

R

S

T

U

V

W

X