

# MACRO for Tera Term

Mar 10, 1998

T. Teranishi

**Copyright (C) 1994-1998 T. Teranishi  
All Rights Reserved.**

MACRO (TTPMACRO.EXE) is an interpreter of the macro language "Tera Term Language (TTL)", which controls Tera Term and provides various functions like auto dialing, auto login and so on.

## **Usage**

[How to run a macro file](#)

[Command line](#)

[How to associate "TTL" files with MACRO](#)

## **Tera Term Language (TTL)**

## **TTL command reference**

## **Appendixes**

### **How to run a macro file**

The executable file TTPMACRO.EXE should be placed in the directory in which TTERMPRO.EXE exists.

There are two ways to run a macro file.

#### **1) From Tera Term.**

To start MACRO, select the [Control] Macro command and then the macro file in the Open Macro dialog box.

#### **2) From MACRO.**

The macro file can be specified as a parameter of the command line (shortcut link) of TTPMACRO.EXE. For example, if you want to run the macro file "DIALUP.TTL", specify the command line (shortcut link) like:

**TTPMACRO DIALUP.TTL**

You can omit the file name extension ".TTL". If you omit the file name, the Open

Macro dialog box appears. It's convenient to install icons (shortcuts) for the macro files you use frequently.

If you choose method 2), you can run Tera Term, after starting the MACRO, by using the "connect" command in the macro file.

While the macro is running, you can pause it, restart it, and stop it by pressing the appropriate buttons in the MACRO dialog box.

## Command line

**TTPMACRO.EXE [/I] [/V] [<macro file> [<second param>] [<third param>]]**

where:

<b>/I</b>	Start MACRO in iconized state.
<b>/V</b>	Start MACRO in hidden (invisible) state.
<b>&lt;macro file&gt;</b>	Macro filename.
<b>&lt;second param&gt;</b>	Character string stored in the system variable "param2".
<b>&lt;third param&gt;</b>	Character string stored in the system variable "param3".

See "Variables" for the system variables "param2" and "param3".

## How to associate "TTL" files with MACRO

To associate the file extension ".TTL" with MACRO, do the following steps.

### a) In Windows 95 or Windows NT 4.0

a-1) Execute the [View] Options command of Explorer.

a-2) Select the "File Types" tab.

a-3) Click the "New Type" button and specify items like the following.

**Description of type: Tera Term macro files**

**Associated extension: TTL**

a-4) Click the "New" button and specify items like the following.

**Action: Execute**

**Application used to perform action:**

**"C:\Program Files\TTERMPRO\ttpmacro.exe" "%1"**

(If Tera Term Pro is installed in C:\Program Files\TTERMPRO.)

a-5) Close all the dialog boxes by clicking "OK" buttons.

**b) In Windows NT 3.51**

b-1) Execute the [File] Associate command of File Manager.

b-2) Specify items like the following and click the "OK" button.

**Files with extension: TTL**

**Associate with:**

**"C:\TTERMPRO\TPMACRO.EXE" "%1"**

(If Tera Term Pro is installed in C:\TTERMPRO.)

## **Tera Term Language (TTL)**

TTL is a simple interpreted language like BASIC. To learn TTL quickly, study the sample macro files in the distribution package and the [command reference](#).

Types

Formats of constants

Identifiers and reserved words

Variables

Expressions and operators

Line formats

## **Types**

TTL have two kinds of data types:

### **Integer**

Signed 16 bit, from -32767 to 32768, in case of TTMACRO.EXE for Windows 3.1.

Signed 32 bit, from -2147483648 to 2147483647, in the case of TTPMACRO.EXE for Windows 95/NT.

### **Character string**

A sequence containing any character except NUL. The maximum length of a string is 255.

## **Formats of constants**

## 1) Integer-type constants

A integer-type constant is expressed as a decimal number or a hexadecimal number which begins with a "\$" character..

Example:

**123**

**-11**

**\$3a**

**\$10F**

Note on negative integer constants

## 2) String-type constants

There are two ways of expressing a string-type constant.

a) A character string quoted by ' or " (both sides must be same).

Example:

**'Hello, world'**

**"I can't do that"**

b) A single character expressed as a "#" followed by an ASCII code (decimal or hexadecimal number). Note: Strings can not contain NUL (ASCII code 0) characters.

Example:

<b>#65</b>	The character "A".
<b>#\$41</b>	The character "A".
<b>#13</b>	The CR character.

ASCII code table

Format a) and b) can be combined in one expression.

Example:

**'cat readme.txt'#13#10**

**abc'#\$0d#\$0a'def'#\$0d#\$0a'ghi'**

## Identifiers and reserved words

### 1) Variable identifiers

The first character must be an alphabetic (A-Z, a-z) or an underscore character "\_". Subsequent characters can be alphabetic, underscore or numeric (0-9). Variable identifiers are not case sensitive. The maximum length is 32.

Example:

**VARIABLE**  
**\_flag**

## 2) Label identifiers

Label identifiers consist of alphabetic, underscore or numeric characters, and are not case sensitive. The maximum length is 32.

Example:

**label1**  
**100**

## 3) Reserved words

The following words are reserved:

[Command]

**bplusrecv, bplussend, changedir...** (see the [command list](#))

[Operator]

**and, not, or, xor**

[System variable]

**inputstr, param2, param3, result, timeout**

## Variables

### 1) User variables

Defined by user. The type of a variable is determined when a value (integer or string) is assigned to it for the first time. Once the type of the variable is determined, values of a different type can not be assigned to it.

### 2) System variables

Each system variable has a predefined type and value. Used with particular

commands.

Variable s	Type	Initial value	Related commands
<b>inputstr</b>	string	""	<a href="#"><u>recvln</u></a> , <a href="#"><u>waitln</u></a> , <a href="#"><u>waitrecv</u></a> , <a href="#"><u>passwordbox</u></a> , <a href="#"><u>inputbox</u></a>
<b>param2</b>	string	*1	*1
<b>param3</b>	string	*1	*1
<b>result</b>	integer	0	<a href="#"><u>bplussend</u></a> , <a href="#"><u>bplusrecv</u></a> , <a href="#"><u>kmtfinish</u></a> , <a href="#"><u>kmtget</u></a> , <a href="#"><u>kmtrecv</u></a> , <a href="#"><u>kmtsend</u></a> , <a href="#"><u>quickvanrecv</u></a> , <a href="#"><u>quickvansend</u></a> , <a href="#"><u>recvln</u></a> , <a href="#"><u>wait</u></a> , <a href="#"><u>waitevent</u></a> , <a href="#"><u>waitln</u></a> , <a href="#"><u>waitrecv</u></a> , <a href="#"><u>xmodemrecv</u></a> , <a href="#"><u>xmodemsend</u></a> , <a href="#"><u>zmodemrecv</u></a> , <a href="#"><u>zmodemsend</u></a> , <a href="#"><u>str2int</u></a> , <a href="#"><u>strcmpare</u></a> , <a href="#"><u>strlen</u></a> , <a href="#"><u>strscan</u></a> , <a href="#"><u>filereadln</u></a> , <a href="#"><u>filesearch</u></a> , <a href="#"><u>filestrseek</u></a> , <a href="#"><u>yesnobox</u></a>
<b>timeout</b>	integer	0	<a href="#"><u>recvln</u></a> , <a href="#"><u>wait</u></a> , <a href="#"><u>waitevent</u></a> , <a href="#"><u>waitln</u></a> , <a href="#"><u>waitrecv</u></a>

\*1 The second and third command line parameter of MACRO. The first parameter is the macro file name. See "[Command line](#)".

## Expressions and operators

Expressions consist of constants, variables, operators, and parentheses. Constants and variables must be of the integer type. The value of an expression is also an integer. The value of a relational expression (formed using relational operators) is 0, if it is true, or 1 if false.

The following are operators:

Category	Precedence	Operators
unary	1, high	<b>not</b>
multiplicative	2	<b>* / %</b>
additive	3	<b>+ - or xor</b>
relational	4, low	<b>= &lt;&gt; &lt; &gt;</b> <b>&lt;= &gt;=</b>

**Note:** the value of expression  $A \% B$  is the remainder of  $A / B$ .

Example:

**1 + 1**

**4 - 2 \* 3**

**15 % 10**

**3 \* (A + 2)**

**A and not B**

**A <= B**

The value is -2.

The value is 5.

A is an integer variable.

A and B are integer variables. The value is 0, if the expression is true, or 1 if false.

## Line formats

There are four kinds of line formats for macro files. Any line can contain a comment which begins with a ";" character. Comments give no effect on the execution of MACRO.

### 1) Empty lines

Lines which have no character or contain only space or tab characters or a comment. They give no effect on the execution of the macro.

Example:

**; Tera Term Language**

### 2) Command lines

Lines containing a single command with parameters.

Format:

**<command> <parameter> ...**

Example:

```

connect'myhost'
wait 'OK' 'ERROR'
if result=2 goto error
sendln 'cat'
pause A*10
end

```

### 3) Assignment lines

Lines which contain an assignment statement.

Format:

**<Variable> = <Value (constant, variable, expression)>**

Example:

**A = 33**

**B = C**

C must already have a value.

**VAL = I\*(I+1)**

**A=B=C**

The value of B=C (0 for false, 1 for true) is assigned to A.

**Error=0<J**

**Username='MYNAME'**

### 4) Label lines

Lines which begin with a ':' character followed by a label identifier.

Format:

**:<Label>**

Example:

**:dial**

**:100**

## TTL command reference

Command index

### Communication commands

bplusrecv      **changed**

bplussend      **changed**

changedir

clearscreen      **new**



<u>closett</u>	
<u>connect</u>	changed
<u>disconnect</u>	
<u>enablekeyb</u>	new
<u>flushrecv</u>	
<u>gettext</u>	
<u>kmtfinish</u>	new
<u>kmtget</u>	new
<u>kmtrecv</u>	changed
<u>kmtsend</u>	changed
<u>loadkeymap</u>	
<u>logclose</u>	
<u>logopen</u>	
<u>logpause</u>	
<u>logstart</u>	
<u>logwrite</u>	
<u>quickvanrecv</u>	changed
<u>quickvansend</u>	changed
<u>recvln</u>	
<u>restoresetup</u>	
<u>send</u>	
<u>sendbreak</u>	
<u>sendfile</u>	
<u>sendkcode</u>	new
<u>sendln</u>	
<u>setecho</u>	
<u>setsync</u>	
<u>settitle</u>	
<u>showtt</u>	changed
<u>testlink</u>	new
<u>unlink</u>	
<u>wait</u>	
<u>waitevent</u>	
<u>waitln</u>	
<u>waitrecv</u>	
<u>xmodemrecv</u>	changed
<u>xmodemsend</u>	changed
<u>zmodemrecv</u>	changed
<u>zmodemsend</u>	changed

## Control commands

<u>call</u>
<u>end</u>

execcmnd  
exit  
for, next  
goto  
if, then, elseif, else, endif  
include  
pause  
return  
while, endwhile

### **String operation commands**

code2str        **new**  
int2str  
str2code        **new**  
str2int  
strcmpare  
strconcat  
strcpy  
strlen  
strscan

### **File operation commands**

fclose  
fileconcat  
filecopy  
filecreate  
filedelete  
filemarkptr    **new**  
fileopen  
filereadln  
filerename  
filesearch  
fileseek  
fileseekback   **new**  
filestrseek  
filestrseek2    **new**  
filewrite  
filewriteln  
findfirst, findnext, findclose   **new**  
getdir        **new**  
makepath       **new**  
setdir        **new**

## Password commands

delpassword  
getpassword  
passwordbox

## Miscellaneous commands

beep  
closesbox  
exec  
getdate  
getenv  
gettime  
inputbox  
messagebox  
setdate  
setdlgpos  
setenv removed  
setexitcode new  
settime  
show  
statusbox  
yesnobox

**bplusrecv** changed

Format:

**bplusrecv**

Causes Tera Term to receive a file from the host with the B-Plus protocol.

Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1.

Otherwise, "result" is set to zero.

**bplussend** changed

Format:

**bplussend <filename>**

Causes Tera Term to send the file <filename> to the host with the B-Plus protocol. Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1.

Otherwise, "result" is set to zero.

Example:

**bplussend 'readme.txt'**

## **changedir**

Format:

**changedir <path>**

Changes the current directory for Tera Term to <path>.

**Note:** the setdir command changes the current directory for MACRO. File names specified for the communication commands (e.g. kmtsend) are based on the current directory for Tera Term. File names specified for other commands (e.g. fileopen) are based on the current directory for MACRO.

Example:

**changedir 'c:\'**

## **clearscreen**

**new**

Format:

**clearscreen <int>**

Causes Tera Term to erase the screen of VT window if <int> is 0.

Causes Tera Term to erase the scroll buffer and screen of VT window if <int> is 1.

Causes Tera Term to erase the screen of TEK window if <int> is 2.

Example:

**clearscreen 0**

## **closett**

Format:

**closett**

Closes Tera Term and enters the unlinked state.

In the unlinked state, the "connect" command can open a new Tera Term window and link MACRO to it.

See also:

"connect"

"disconnect"

"testlink"  
"unlink"

Example:

**closett**  
**connect 'host'**

**connect**            **changed**

Format:

**connect <command line parameters>**

If MACRO is not linked to Tera Term, this command runs Tera Term with <command line parameters>, and links it to MACRO.

If MACRO has already been linked to Tera Term and Tera Term is not connected to the host, this command causes Tera Term to connect to the host specified by <command line parameters>.

See Tera Term help for the format of <command line parameters>.

If MACRO has already been linked to Tera Term and Tera Term has already been connected to the host, this command is ignored.

As a result of this command, the system variable "result" is set to one of the following values depending on the link and connection status:

Value	Status
0	Link to Tera Term has not been made.
1	Connection to the host has not been made, but link to Tera Term has been made.
2	Both connection and link have been made.

To test the current link and connect status before executing the "connect" command, use the "testlink" command.

Communication commands except "connect" and "testlink" can not be executed before the link is established.

See also:

"closett"

"disconnect"

"testlink"

"unlink"

Example:

**connect "**

No command line  
parameter

**connect '/C=2'**

Run Tera Term with  
parameter "/C=2"

**connect 'foohost.foo.foo.jp'**

**CommandLine =**

**'111.111.11.11'**

**connect CommandLine**

## **disconnect**

Format:

**disconnect**

Closes the communication between Tera Term and the host.

If Tera Term is not terminated by this command, the link between Tera Term and MACRO is kept.

See also:

"closett"

"connect"

"testlink"

"unlink"

## **enablekeyb**

**new**

Format:

**enablekeyb <flag>**

Enables or disables keyboard input of Tera Term. The value of <flag> should be 1 for enabling and 0 for disabling.

Example:

**enablekeyb 0**

## **flushrecv**

Format:

**flushrecv**

Clears received characters in the buffer of MACRO.

Characters received from the host are transferred to MACRO. MACRO stores the characters in the buffer. Character-reading commands, such as the "wait" command, read out them from the buffer. Characters in the buffer are kept until character-reading commands process them or the buffer overflows or the flushrecv command clears the buffer.

The "flushrecv" command can be used to avoid unexpected results of character-reading commands caused by old characters in the buffer.

## **gettext**

Format:

**gettext <strvar>**

Retrieves the title text of Tera Term and stores it in the string variable <strvar>.

Example:

gettext titletext

## **kmtfinish**      **new**

Format:

**kmtfinish**

Causes Tera Term to execute the Kermit Finish command.

Pauses until the end of the Finish command.

If the command is executed successfully, the system variable "result" is set to 1. Otherwise, "result" is set to zero.

## **kmtget**      **new**

Format:

**kmtget <filename>**

Causes Tera Term to get the file <filename> from the host by using the Kermit Get command. The host should be in the server state. Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1. Otherwise, "result" is set to zero.

Example:

**kmtget '\*.\***

**kmtrecv**                      **changed**

Format:

**kmtrecv**

Causes Tera Term to receive a file from the host with the Kermit protocol. Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1. Otherwise, "result" is set to zero.

**kmtsend**                      **changed**

Format:

**kmtsend <filename>**

Causes Tera Term to send the file <filename> to the host with the Kermit protocol. Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1. Otherwise, "result" is set to zero.

Example:

**kmtsend 'readme.txt'**

**loadkeymap**

Format:

**loadkeymap <filename>**

Causes Tera Term to load a keyboard setup file specified by <filename>.



Example:

```
loadkeymap 'keyboard.cnf'
```

## **logclose**

Format:

```
logclose
```

Causes Tera Term to close the log file.

## **logopen**

Format:

```
logopen <filename> <binary flag> <append flag>
```

Causes Tera Term to start logging. Received characters are written to the file <filename>.

If <binary flag> is zero, received new-line characters are converted (CR -> CR/CRLF) and escape sequences are stripped out. If <binary flag> is non-zero, received characters are written without any modifications.

If <append flag> is non-zero and the file <filename> already exists, received characters are appended to it. If <append flag> is zero and the file <filename> already exists, the file is overwritten.

Example:

```
logopen 'myhost.log' 0 0
```

## **logpause**

Format:

```
logpause
```

Causes Tera Term to pause logging. Received characters are discarded while logging is paused.

## **logstart**

Format:

```
logstart
```

Causes Tera Term to restart the logging, if paused.

## logwrite

Format:

**logwrite <string>**

Appends a <string> to the log file of the Tera Term.

This command is valid only while Tera Term is logging. The <string> can be written even while logging is paused.

Example:

**logwrite 'LOG FILE'#13#10**

## quickvanrecv **changed**

Format:

**quickvanrecv**

Causes Tera Term to receive a file from the host with the Quick-VAN protocol.

Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1.

Otherwise, "result" is set to zero.

## quickvansend **changed**

Format:

**quickvansend <filename>**

Causes Tera Term to send the file <filename> to the host with the Quick-VAN protocol. Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1.

Otherwise, "result" is set to zero.

Example:

**quickvansend 'readme.txt'**

## recvln

Format:

**recvln**

Retrieves a line of received characters from the host and stores it in the system

variable "inputstr".

This command waits until a line is received or the communication between Tera Term and the host is terminated or the timeout occurs. If the system variable "timeout" is greater than zero, the timeout occurs when <timeout> seconds have passed. If the "timeout" is less than or equal to zero, the timeout never occurs.

If the line is received successfully, the system variable "result" is set to 1. Otherwise, "result" is set to zero.

Example:

<b>fileopen file 'log.txt' 0</b>	open the log file
<b>setsync 1</b>	enter synchronous mode
<b>result=1</b>	
<b>while result=1</b>	
<b>recvln</b>	receive one line
<b>filewriteln file inputstr</b>	write it to the log file
<b>endwhile</b>	
<b>setsync 0</b>	enter asynchronous mode

See also "[setsync](#)" for the synchronous mode.

## **restoresetup**

Format:

**restoresetup <filename>**

Causes Tera Term to load a Tera Term setup file specified by <filename>.

Example:

**restoresetup 'teraterm.ini'**

## **send**

Format:

**send <data1> <data2> ....**

Causes Tera Term to send characters to the host.

If <data> is a string, the string is sent to the host. If <data> is an integer, its lowest-order byte (0-255) is regarded as an ASCII code of the character, and the character is sent to the host.

Example:

**send 'ABC'**

**send 65 66 67**

Send "ABC". (ASCII  
code of the character  
"A" is 65.)

**myname='Tera Term'**

**send 'My name is ' myname .'**

## **sendbreak**

Format:

**sendbreak**

Causes Tera Term to send a break signal to the host.

## **sendfile**

Format:

**sendfile <filename> <binary flag>**

Causes Tera Term to send the file <filename> to the host. Pauses until the end of the file transfer.

If <binary flag> is non-zero, the file is sent without any modifications.

If <binary flag> is zero, new-line characters are converted (CR -> CR/CRLF) and control characters except TAB, LF and CR are stripped out.

Example:

**sendfile 'data.dat' 1**

## **sendkcode**                    **new**

Format:

**senkcode <key code> <repeat count>**

Causes Tera Term to perform a function defined for pressing a key or key combination. The key or key combination is specified by its <key code>, which is defined by KEYCODE.EXE. The function is performed <repeat count> times.

Usually the function means sending a character string to the host. The function can be defined in the keyboard setup file of Tera Term. See KEYCODE.TXT.

Example:

**sendkcode 336 3**

Simulate pressing the down-arrow key three times. The key code of the down-arrow key is 336 for the IBM PC keyboard.

## **sendln**

Format:

**sendln <data1> <data2> ....**

Causes Tera Term to send characters followed by a new-line character to the host.

Format of <data> is the same as the "send" command.

Example:

**sendln**

Only a new-line character is sent.

**sendln 'abc'**

**Password='mypassword'**  
**sendln Password**

## **setecho**

Format:

**setecho <echo flag>**

Changes the local echo status of Tera Term.

If <echo flag> is non-zero, the local echo is turned on.

If <echo flag> is zero, the local echo is turned off.

Example:

**setecho 1**

local echo on

## **setsync**

Format:

## **setsync <sync flag>**

Enters the synchronous communication mode if <sync flag> is non-zero, or enters the asynchronous communication mode if <sync flag> is zero.

Tera Term transfers received characters from the host to MACRO. MACRO stores the characters in the buffer. The character-reading commands, such as the "wait" command, read out the characters from the buffer.

Initially, MACRO is in the asynchronous mode. In this mode, the buffer may overflow if no character-reading command is executed for a long time, or the receiving speed is too fast.

In the synchronous mode, the buffer never overflows. If the buffer becomes full, Tera Term stops receiving characters from the host and stops transferring them to MACRO. When the buffer regains enough space, Tera Term restarts receiving and transferring.

Enter the synchronous mode only when it is necessary and re-enter the asynchronous mode when the synchronous operation is no longer needed.

For a macro operation which requires reliability, something like processing lines of received characters without loss of data, you need to enter the synchronous mode. However, the synchronous mode makes Tera Term slow in speed of receiving characters and causes Tera Term freeze if no character-reading command is executed for a long time. On the other hand, a simple macro operation, such as auto login, works with almost no problem in the asynchronous mode, because the buffer size is large enough (4096 bytes) and all received characters are processed by character-reading commands before the buffer overflows.

See also "flushrecv" for clearing the buffer.

Example:

<b>setsync 1</b>	enter the synchronous mode
<b>setsync 0</b>	enter the asynchronous mode

## **settitle**

Format:

**settitle <title>**

Changes the title text of Tera Term to <title>.

Example:

**settitle 'Tera Term'**

**showtt**                      **changed**

Format:

**showtt <show flag>**

Hides the VT window of Tera Term if <show flag> is -1.

Minimizes the VT window of Tera Term if <show flag> 0.

Restores the VT window Tera Term if <show flag> is 1.

Hides the TEK window of Tera Term if <show flag> is 2.

Minimizes the TEK window of Tera Term if <show flag> 3.

Open/restores the TEK window of Tera Term if <show flag> is 4.

Closes the TEK window of Tera Term if <show flag> is 5.

Example:

**showtt 0**

Minimize Tera Term.

**showtt 1**

Restore Tera Term.

**showtt -1**

Hide Tera Term.

**testlink**                      **new**

Format:

**testlink**

Reports the current link and connection status.

The system variable "result" is set to one of the following values depending on the link and connection status:

Value	Status
0	Link to Tera Term has not been made.
1	Connection to the host has not been made, but link to Tera Term has been made.

2

Both connection  
and link have been  
made.

See also:

"closett"

"connect"

"disconnect"

"unlink"

Example:

**testlink**

**if result=0 connect 'host'**

If MACRO is not linked  
to Tera Term, execute  
the connect command.

## unlink

Format:

**unlink**

Terminates the link between the current Tera Term window and MACRO.  
MACRO enters the unlinked state and can not controll the Tera Term window  
any more.

In the unlinked state, the "connect" command can open a new Tera Term window  
and link MACRO to it.

See also:

"closett"

"connect"

"disconnect"

"testlink"

Example:

**connect 'host1'**

**unlink**

**connect 'host2'**

open a Tera Term  
window and link  
MACRO to it  
terminate the link  
open another Tera Term  
window and link  
MACRO to it



## wait

Format:

**wait <string1> <string2> ...**

Pauses until one of the character strings is received from the host, or until the timeout occurs. Maximum number of the strings is 10.

If the system variable "timeout" is greater than zero, the timeout occurs when <timeout> seconds have passed. If the "timeout" is less than or equal to zero, the timeout never occurs.

The "wait" command returns one of the following values in the system variable "result":

Value	Meaning
0	Timeout. No string has received.
1	<string1> has received.
2	<string2> has received.
.	.
.	.

Example:

<b>timeout = 30</b>	The timeout limit is 30 sec.
<b>Wait 'OK' 'ERROR'</b>	
<b>if result=0 goto timeout</b>	If timeout occurs, go to ":timeout".
<b>If result=1 goto ok</b>	If "OK" has received, go to ":error".
<b>If result=2 goto error</b>	If "ERROR" has received, go to ":error".
<b>wait #10'&gt;' 'complete.'#13</b>	Wait a line beginning with the ">" or a line ending with the "complete.". (ASCII code of LF is 10, and CR is 13.)

## waitevent

Format:

**waitevent <events>**

Pauses until one of the events specified by <events> occurs.

<events> can be combination of the following event identifiers.

Event	Event identifier
timeout	1
unlink	2
disconnection	4
connection	8

The timeout event occurs when <timeout> seconds have passed.

<timeout> is the value of the system variable "timeout".

If <timeout> is less than or equal to zero, this event never occurs.

The unlink event occurs when Tera Term is closed.

The disconnection (connection) event occurs when the communication between Tera Term and the host is closed (opened).

The "waitevent" command returns the identifier of the actual event in the system variable "result".

Example:

<b>waitevent 4</b>	Wait the disconnection event
<b>waitevent 2 or 8</b>	Wait the unlink or connection events
<b>if result=2 goto label1</b>	The unlink event occurred
<b>if result=8 goto label2</b>	The connection event occurred

## waitln

Format:

**waitln <string1> <string2> ...**

Pauses until a line which contains one of the character strings is received from the host, or until the timeout occurs. Maximum number of the strings is 10.

If the system variable "timeout" is greater than zero, the timeout occurs when <timeout> seconds have passed. If the "timeout" is less than or equal to zero, the timeout never occurs.

The "waitln" command returns the received line in the system variable "inputstr" and one of the following values in the system variable "result":

Value	Meaning
0	Timeout.
1	A line which contains <string1> has received.
2	A line which contains <string2> has received.
.	.
.	.

## **waitrcv**

Format:

**waitrcv <sub-string> <len> <pos>**

Pauses until a string, which satisfies a condition, is received from the host, or until the timeout occurs.

The condition is:

The length of the string is <len>, and the string contains the <sub-string> beginning at the <pos>th character.

For example, if <sub-string> is "def" and <len> is 9 and <pos> is 4, the string "abcdefghi" satisfies the condition.

If such a string is received, it is saved in the system variable "inputstr".

If the system variable "timeout" is greater than zero, the timeout occurs when <timeout> seconds have passed. If the "timeout" is less than or equal to zero, the timeout never occurs.

The "waitrecv" command returns one of the following values in the system variable "result":

Value	Meaning
-1	A string, which contains the <sub-string> beginning at the <pos>th character, has been received, and saved in the "inputstr", but its length is less than <len> because of the timeout.
0	Timeout. No string, which satisfies the condition, has been received.
1	A string, which satisfies the condition, has been received, and saved in the "inputstr".

## **xmodemrecv**      **changed**

Format:

**xmodemrecv <filename> <binary flag> <option>**

Causes Tera Term to receive the file <filename> from the host with the XMODEM protocol. Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1. Otherwise, "result" is set to zero.

If the file is a binary file, <binary flag> must be non-zero. If the file is a text file, <binary flag> must be zero.

<option> specifies the XMODEM option, and can be one of the following:

<option> >	XMODEM option
1	Checksum
2	CRC
3	1K
others	Checksum

Example:

**xmodemrecv 'readme.txt' 0 2**      XMODEM receive, text  
file, CRC

## **xmodemsend** **changed**

Format:

**xmodemsend <filename> <option>**

Causes Tera Term to send the file <filename> to the host with the XMODEM protocol. Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1.

Otherwise, "result" is set to zero.

<option> specifies the XMODEM option, and can be one of the following:

<option> >	XMODEM option
1	Checksum
2	CRC
3	1K
others	Checksum

Example:

**xmodemsend 'readme.txt' 1**      XMODEM send,  
checksum

## **zmodemrecv** **changed**

Format:

**zmodemrecv**

Causes Tera Term to receive files from the host with the ZMODEM protocol.

Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1.

Otherwise, "result" is set to zero.

## **zmodemsend** **changed**

Format:

**zmodemsend <filename> <binary flag>**

Causes Tera Term to send the file <filename> to the host with the ZMODEM protocol. Pauses until the end of the file transfer.

If the file is transferred successfully, the system variable "result" is set to 1.

Otherwise, "result" is set to zero.

If the file is a binary file, <binary flag> must be non-zero. If the file is a text file, <binary flag> must be zero.

Example:

```
zmodem 'readme.txt' 0
```

## **call**

Format:

```
call <label>
```

Calls a subroutine beginning with the <label> line.

Example:

```
messagebox "I'm in main."  
"test"  
call sub                                Jump to ":sub".  
messagebox "Now I'm in  
main" "test"  
end  
  
:sub                                    Start of the subroutine.  
    messagebox "Now I'm in  
sub" "test"  
    return                            Go back to the main  
                                        routine.
```

## **end**

Format:

```
end
```

Quits the execution of the macro. MACRO is also closed.

## **execcmnd**

Format:

```
execcmnd <statement>
```

Executes a TTL statement expressed by the string <statement>.

Example:

**execcmnd "send 'abc'"**

Execute the statement  
"send 'abc'".

**execcmnd "a=1"**

## **exit**

Format:

**exit**

Exits the include file and returns to the main file.

Example:

See "include".

## **for, next**

Format:

**for <intvar> <first> <last>**

**...**

**...**

**next**

Repeats the statements between "for" and "next" until the integer variable <intvar> has the value <last> at the 'next' statement.

The initial value of the <intvar> is <first>. If <last> is greater than <first>, <intvar> is incremented by 1 at the 'next' line. If <last> is less than <first>, <intvar> is decremented by 1 at the 'next' line.

Example:

**for i 1 10**  
**sendln 'abc'**  
**next**

Repeat ten times.

**for i 5 1**  
**sendln 'abc'**  
**next**

Repeat five times.

## **goto**

Format:

**goto <label>**

Moves control to the next line of the <label>.

Example:

```
goto label
```

Jump to the next line of the ':label'.

```
...
```

```
...
```

```
...
```

```
:label
```

```
send 'abc'
```

## **if, then, elseif, else, endif**

1) Format:

```
if <int> <statement>
```

Executes a <statement>, if <int> is non-zero.

Example:

```
if A>1 goto label
```

If A>1, jump to ':label'.

```
if result A=0
```

If result<>0, assign 0 to A.

2) Format:

```
if <int 1> then
```

```
...
```

```
(Statements for the case: <int 1> is true (non-zero).)
```

```
...
```

```
[elseif <int 2> then]
```

```
...
```

```
(Statements for the case: <int 1> is false (zero) and <int 2> is true.)
```

```
...
```

```
[elseif <int N> then]
```

```
...
```

```
(Statements for the case: <int 1>, <int 2>,... and <int N-1> are all false, and <int N> is true.)
```

```
...
```

```
[else]
```

```
...
```

```
(Statements for the case: all the conditions above are false (zero).)
```

```
...
```



## **endif**

'if' and 'elseif' statements must end with 'then'.

'elseif' and 'else' can be omitted.

'endif' can not be omitted.

Examples:

```
if a=1 then  
  b = 1  
  c = 2  
  d = 3  
endif
```

```
if i<0 then  
  i=0  
else  
  i=i+1  
endif
```

```
if i=1 then  
  c = '1'  
elseif i=2 then  
  c = '2'  
elseif i=3 then  
  c = '3'  
else  
  c = '?'  
endif
```

## **include**

Format:

```
include <include file name>
```

Moves control to the include file.

Example:

----- Main file "main.ttl" -----

```
  i=10  
  :loop  
  include 'sub.ttl'  
  if i>=0 goto loop  
  end
```

Move to the include file.

----- End of "main.ttl" -----

----- Include file "sub.ttl" -----

**if i<0 then**

**messagebox 'error!' 'sub'**

**exit**

Go back to the main file.

**endif**

**i = i - 1**

Go back to the main file.

----- End of "sub.ttl" -----

## pause

Format:

**pause <time>**

Pauses for <time> seconds.

Example:

**pause 10**

Pause for 10 seconds.

**pause Time**

## return

Format:

**return**

Exits the subroutine and returns to the main routine.

Example:

See "call".

## while, endwhile

Format:

**while <int>**

...

...

...

**endwhile**

Repeats the statements between 'while' and 'endwhile' while <int> is non-zero.

Examples:

```
i = 10
while i>0
    i = i - 1
endwhile
```

Repeat ten times.

## **code2str**      **new**

Format:

**code2str <strvar> <ASCII code>**

If the integer value <ASCII code> is 1-255, this command copies a character with the <ASCII code> to the string variable <strvar>.

This command converts an ASCII code sequence expressed by <ASCII code> to a character string and copies it to <strvar>. The non-zero highest-order byte of <ASCII code> is regarded as the first byte of the ASCII code sequence. If <ASCII code> is zero, <strvar>.is set to null (""). The maximum length of the character string is 2 for TTMACRO.EXE and 4 for TTPMACRO.EXE.

Example:

**code2str str \$41**

The character "A" is stored in the variable "str". The ASCII code for "A" is \$41.

**code2str str \$4142**

The character string "AB" is stored in the variable "str". The ASCII code \$41 is for "A" and \$42 for "B".

## **int2str**

Format:

**int2str <strvar> <integer value>**

Converts <integer value> to its string expression, and returns it in the string variable <strvar>.

Example:

**int2str valstr 123**

The string "123" is assigned to the variable "valstr".

## **str2code**      **new**

Format:

**str2code <intvar> <string>**

If the <string> consists of one character, this function copies the ASCII code for the character to the integer variable <intvar>.

If the length of <string> is longer than one, this function converts <string> to its ASCII code sequence and copies it to <intvar>. The variable <intvar> can store n ASCII codes at maximum, where n is 2 for TTMACRO.EXE and 4 for TTPMACRO.EXE. If the length of <string> is longer than n, the last n bytes of ASCII code sequence is copied to <intvar>.

Example:

<b>str2code val 'A'</b>	val=65 (ASCII code for "A")
<b>str2code val 'AB'</b>	val=65*256+66

## **str2int**

Format:

**str2int <intvar> <string>**

Converts the <string> which represents a decimal number to its numeric value. The value is returned in the integer variable <intvar>. If the string is converted successfully, the system variable "result" is set to 1. Otherwise, "result" is set to zero.

Example:

<b>str2int val '123'</b>	val=123, result=1
<b>str2int val '123abc'</b>	result=0

## **strcompare**

Format:

**strcompare <string1> <string2>**

Compares two strings. Depending on the relation between them, one of the following result code is returned in the system variable "result":

Relation	Result
<string1> < <string2>	-1
<string1> = <string2>	0
<string1> > <string2>	1

Example:

```
strcmpare 'abc' 'def'          result = -1
```

```
strcmpare command 'next'
if result=0 goto label
strcmpare command 'end'
if result=0 end
```

## strconcat

Format:

```
strconcat <strvar> <string>
```

Appends a copy of <string> to the end of the string variable <strvar>.

Example:

```
filename = 'c:\teraterm\'
strconcat filename 'test.txt'
```

## strcopy

Format:

```
strcopy <string> <pos> <len> <strvar>
```

Copies a substring of <string> to the string variable <strvar>.

The substring begins at the <pos>th character in <string>, and its length is <len>.

Example:

```
strcopy 'tera term' 6 4 substr    substr='term'
```

## strlen

Format:

```
strlen <string>
```

Returns the length of <string> in the system variable "result".

Example:

**strlen 'abc'**

result = 3

## **strscan**

Format:

**strscan <string> <substring>**

Searches for <substring> in <string>.

If <substring> is found, its position is returned in the system variable "result". If <string> contains more than one occurrence of <substring>, the position of the first one is returned. If <substring> is not found, "result" is set to zero.

Example:

**strscan 'tera term' 'term'**

result = 6

## **fileclose**

Format:

**fileclose <file handle>**

Closes the file specified by <file handle>.

<file handle> is no longer valid after this command.

Example:

**fileclose fhandle**

## **fileconcat**

Format:

**fileconcat <file1> <file2>**

Appends a copy of file <file2> to the end of file <file1>.

<file1> and <file2> must not be same.

Example:

**fileconcat 'test.dat' test2.dat'**

## **filecopy**

Format:

**filecopy <file1> <file2>**

Copies file <file1> to file <file2>.

If <file2> already exists, it is overwritten. <file1> and <file2> must not be same.

Example:

```
filecopy 'test.dat' test2.dat'
```

## **filecreate**

Format:

```
filecreate <file handle> <filename>
```

Creates and opens a new file specified by <filename>.

The file pointer is set to the beginning of the file. If file <filename> already exists, its size is truncated to zero. If the file is successfully created and opened, the file handle is returned in the integer variable <file handle>. Otherwise, <file handle> is set to -1.

Example:

```
filecreate fhandle 'data.dat'
```

## **filedelete**

Format:

```
filedelete <filename>
```

Deletes the file specified by <filename>.

Example:

```
filedelete 'temp.log'
```

## **filemarkptr** **new**

Format:

```
filemarkptr <file handle>
```

Marks the current file pointer for an opened file specified by <file handle>. The marked pointer can be recalled by the "fileseekback" command.

Example:

```
filemarkptr fhandle
```

See also the example of "filestrseek2" command.

## fileopen

Format:

**fileopen <file handle> <file name> <append flag>**

Opens a file specified by <file name>.

If the file does not exist, it is created and then opened. If the file is successfully opened, the file handle is returned in the integer variable <file handle>.

Otherwise, <file handle> is set to -1.

If <append flag> is zero, the file pointer is set to the beginning of the file. If <append flag> is non-zero, the file pointer is set to the end of the file.

Example:

**fileopen fhandle 'data.dat' 0**

**fileopen fhandle 'data.dat' 1**

## filereadln

Format:

**filereadln <file handle> <strvar>**

Reads a line from the file specified by <file handle>.

The line is written into the string variable <strvar>. The file pointer is moved to the beginning of the next line. If the file pointer reaches the end of the file while reading the line, the system variable "result" is set to 1. Otherwise, "result" is set to zero.

Example:

<b>fileopen fhandle 'test.txt' 0</b>	Open a file.
<b>:loop</b>	
<b>filereadln fhandle line</b>	Read a line from the file.
<b>if result goto fclose</b>	
<b>messagebox line 'test.txt'</b>	Display the line.
<b>goto loop</b>	Repeat until the end of the file.
<b>:fclose</b>	
<b>fileclose fhandle</b>	Close the file.

## filerename



Format:

**filerename <file1> <file2>**

Renames <file1> to <file2>.

<file1> and <file2> must not be same.

Example:

**filerename 'test.dat' test2.dat'**

## filesearch

Format:

**filesearch <filename>**

Searches for the file specified by <filename>.

If it is found, the system variable "result" is set to 1. Otherwise, "result" is set to zero.

Example:

**filesearch 'readme.txt'**  
**if result=0 messagebox 'File not found.' 'error'**

## fileseek

Format:

**fileseek <file handle> <offset> <origin>**

Moves the pointer for the file specified by <file handle>.

With this command, the file pointer is moved <offset> bytes from:

the beginning of the file, if <origin> is 0.

the current position, if <origin> is 1.

the end of the file, if <origin> is 2.

Example:

**fileseek fhandle 0 0**

Move to the beginning of file.

**fileseek fhandle 10 1**

Move 10 bytes from the current position.

**fileseek fhandle 0 2**  
**fileseek fhandle 0-10 2**

Move to the end of file.  
Move to the position 10  
bytes backward from the  
end of file.

## **fileseekback** new

Format:

**fileseekback <file handle>**

Moves the file pointer for an opened file to the position marked by the "filemarkptr" command. The file is specified by <file handle>.

Example:

**fileseekback fhandle**  
See also the example of "filestrseek2" command.

## **filestrseek**

Format:

**filestrseek <file handle> <string>**

Searches for <string> in the file specified by <file handle>.  
The search is done forward and started from the current position of the file pointer.  
For the backward search, use the "filestrseek2" command.

If <string> is found, the file pointer is moved to the next character of the string and the system variable "result" is set to 1. If <string> is not found, the file pointer is not moved and "result" is set to zero.

Example:

**fileopen fhandle 'teraterm.log'**  
**0**

**filestrseek fhandle 'abc'**

Search for the string  
"abc" in the file  
"teraterm.log".

**if result=0 goto not\_found**  
**filereadln fhandle str**

Read characters from  
the next of "abc" to the  
end of the line.

**:not\_found**  
**fileclose fhandle**

## filestrseek2

new

Format:

**filestrseek2 <file handle> <string>**

Searches for <string> in the file specified by <file handle>.

The search is done backward and started from the current position of the file pointer.

For the forward search, use the "filestrseek" command.

If <string> is found, the file pointer is moved to the position of the character just before the string and the system variable "result" is set to 1. If <string> is not found, the file pointer is not moved and "result" is set to zero. If the file pointer is already zero before the execution of this command, "result" is set to zero.

Example:

<b>fileopen fhandle 'teraterm.log'</b>	Open the file
<b>1</b>	"teraterm.log". The file pointer is set to the end of file.
<b>:next</b>	
<b>filestrseek2 fhandle 'abc'</b>	Backward search for the string "abc"
<b>if result=0 goto not_found</b>	
<b>filemarkptr fhandle</b>	mark the file pointer
<b>filereadln fhandle str</b>	Read characters from the character just before "abc" to the end of the line.
<b>fileseekback fhandle</b>	recall the file pointer
<b>goto next</b>	search for the next word
<b>:not_found</b>	
<b>fileclose fhandle</b>	

## filewrite

Format:

**filewrite <file handle> <string>**

Writes <string> to the file specified by <file handle>.

Example:

**filewrite fhandle '-----cut here-----' #13#10**

## **filewriteln**

Format:

**filewriteln <file handle> <string>**

Writes <string> and the new-line characters (CR+LF) to the file specified by <file handle>.

Example:

**filewriteln fhandle '-----cut here-----'**

## **findfirst**                      **new**

Format:

**findfirst <dir handle> <file name> <strvar>**

The findfirst command searches for the first file matching the specified file name <file name>. If the file is found, this command returns the directory handle in <dir handle>, returns the first file name in <strvar> and sets the system variable "result" to 1. Otherwise, <dir handle>, <strvar> and "result" are set to -1, "" and 0, respectively.

If the findfirst command find the first file successfully, the directory handle can be used for the findnext command to search for the next file matching the specified <file name>. **The directory handle should be released by the findclose command.**

## **findnext**                      **new**

Format:

**findnext <dir handle> <strvar>**

The findnext command searches for the next file matching the file name specified in the findfirst command. The integer value <dir handle> should be the directory handle returned by the findfirst command. If the next file is found, the file name is returned in <strvar> and "result" is set to 1. Otherwise, <strvar> and "result" are set to "" and 0, respectively.

## **findclose**                      **new**

Format:

## **findclose <dir handle>**

The findclose command releases the directory handle <dir handle> returned by the findfirst command. This command should be executed if the findfirst command is executed successfully.

Example:

```
findfirst dh '*.txt' filename
while result
  messagebox filename '*.txt'
  findnext dh filename
endwhile
findclose dh
```

## **getdir new**

Format:

```
getdir <strvar>
```

Returns the current working directory for MACRO (not for Tera Term) to the string variable <strvar>.

See also:

"setdir"

Example:

```
getdir dir
```

## **makepath new**

Format:

```
makepath <strvar> <dir> <name>
```

Creates the full path name from the directory name <dir> and file name <name>. The full path name is stored in the string variable <strvar>. If necessary, '\' is inserted between <dir> and <name>.

Example:

```
makepath path 'c:\teraterm'    path =
'test.txt'                    "c:\teraterm\test.txt"
```

## **setdir new**

Format:

**setdir <dir>**

Changes the current working directory for MACRO to <dir>.

**Note:** the changedir command changes the current directory for Tera Term. File names specified for the communication commands (e.g. kmtsend) are based on the current directory for Tera Term. File names specified for other commands (e.g. fileopen) are based on the current directory for MACRO.

Example:

**setdir 'c:\'**

## delpassword

Format:

**delpassword <filename> <password name>**

Deletes a password specified by <password name> in the password file <filename>. If <password name> is a blank string, all passwords in the file are deleted.

See "getpassword" for the password file.

Example:

**delpassword 'password.dat' 'mypassword'**

## getpassword

Format:

**getpassword <filename> <password name> <strvar>**

Retrieves an encrypted password identified by <password name> from the password file <filename>. Decrypts the password and stores it into the string variable <strvar>.

If the specified file does not exist, it is newly created.

If the specified password is not stored in the file, the password dialog box appears and the entered password is stored in <strvar>. At the same time, the new password is encrypted and written in the file with the identifier <password name>.

A password file can contain multiple passwords. Each of them is identified by the

password identifier.

Example:

```
getpassword 'password.dat' 'mypassword' password  
connect 'myhost'  
wait 'login:'  
sendln 'myname'  
wait 'password:'  
sendln password
```

## passwordbox

Format:

```
passwordbox <message> <title>
```

Displays a dialog box prompting the user to input a password.

The <message> is displayed in the dialog box. The <title> is displayed as the dialog box title. The password typed by the user is not displayed as is. Instead, asterisks are displayed. The password is returned in the system variable "inputstr".

Example:

```
passwordbox 'Enter password' 'Login'
```

## beep

Format:

```
beep
```

Makes a beep sound.

## closesbox

Format:

```
closesbox
```

Closes the status dialog box opened by the "statusbox" command.

Example:

See "statusbox".

## exec

Format:

**exec <command line>**

Runs an application specified by <command line>.

Format:

**exec 'notepad readme.txt'**      Run "Notepad".

## **getdate**

Format:

**getdate <strvar>**

Returns the current date in the string variable <strvar>, with the format "YYYY-MM-DD".

Example:

**getdate datestr**

## **getenv**

Format:

**getenv <envname> <strvar>**

Retrieves the value of an environment variable specified by <envname> and stores it in the string variable <strvar>.

Example:

**getenv 'TEMP' env**

## **gettime**

Format:

**gettime <strvar>**

Returns the current time in the string variable <strvar>, with the format "HH:MM:SS".

Example:

**gettime timestr**

## **inputbox**



Format:

**inputbox <message> <title>**

Displays a dialog box prompting user to input a string.

The <message> is displayed in the dialog box. The <title> is displayed as the dialog box title. The string entered by the user is returned in the system variable "inputstr".

Example:

**inputbox 'Password:' 'Login'**  
**sendln inputstr**

## **messagebox**

Format:

**messagebox <message> <title>**

Displays a dialog box with <message> and <title>.

Example:

**messagebox ErrorMessage 'Error'**

## **setdate**

Format:

**setdate <date>**

Sets the system date to <date>. The format of <date> should be "YYYY-MM-DD".

Example:

**setdate '1997-06-30'**

## **setdlgpos**

Format:

**setdlgpos <x> <y>**

Changes the initial position for dialog boxes opened by the "inputbox", "messagebox", "passwordbox" and "statusbox" commands. If the status dialog box is displayed, the "setdlgpos" command also moves the dialog box.

<x> and <y> specify the position (x,y) in the screen coordinate.  
The origin (0,0) is upper left corner of the screen.

Example:

<b>setdlgpos 0 0</b>	
<b>messagebox 'Message' 'Title'</b>	Message box at the upper left corner.
<b>setdlgpos 0 200</b>	Open the status box.
<b>statusbox 'Message' 'Title'</b>	
<b>for i 0 200</b>	
<b>setdlgpos i 200</b>	Moves the status box.
<b>next</b>	

**setexitcode**                      **new**

Format:

**setexitcode <exit code>**

Sets the exit code of MACRO to the integer value <exit code>.

**For Windows 3.1:** There is no way to utilize the exit code of MACRO because MACRO can not be run by a batch file. This command is provided just for compatibility with the 32-bit version of MACRO.

**For Windows 95:** If MACRO is run from a batch file by the command line "start /w ttpmacro <tfl filename>" (the option /w is necessary), the exit code can be tested by the DOS command "if errorlevel n".

**For Windows NT:** If MACRO is run from a batch file by the command line "ttpmacro <tfl filename>", the exit code can be tested by the DOS command "if errorlevel n".

Example:

----- Batch file "test.bat" for Win 95 -----

**start /w ttpmacro test.ttl**

**if errorlevel 1 echo Error!**

----- End of "test.bat" -----

Run MACRO using the "start /w" command.  
Display message if the exit code is greater than 0.

----- Batch file "test.bat" for Win NT -----

**ttmacro test.ttl**

**if errorlevel 1 echo Error!**

----- End of "test.bat" -----

----- Macro file "test.ttl" -----

**setexitcode 1**

----- End of "test.ttl" -----

Run MACRO. No need to use the "start" command.

Display message if the exit code is greater than 0.

Set the exit code to 1.

## settime

Format:

**settime <time>**

Sets the system time to <time>. The format of <time> should be "HH:MM:SS".

Example:

**settime '01:05:00'**

## show

Format:

**show <show flag>**

Minimizes MACRO, if <show flag> is zero.

Restores MACRO, if <show flag> is greater than zero.

Hides MACRO, if <show flag> is less than zero.

Example:

**show 0**

**show 1**

**show -1**

Minimize MACRO.

Restore MACRO.

Hide MACRO.

## statusbox

Format:

**statusbox <message> <title>**

Displays the status dialog box if it has not been displayed yet.

Changes the message to <message> and title to <title>.

The "setdlgpos" command changes the position of status dialog box.

The "closesbox" command closes the status dialog box.

Example:

<b>setdlgpos 200 200</b>	Set the initial position.
<b>statusbox 'Message' 'Title'</b>	Display the status dialog box.
<b>pause 3</b>	
<b>setdlgpos 0 0</b>	Move the dialog box.
<b>pause 3</b>	
<b>closesbox</b>	Close the dialog box.

## yesnobox

Format:

**yesnobox <message> <title>**

Displays a dialog box with the <message>, <title>, "Yes" button and "No" button.

If the user clicks on the "Yes" button, the system variable "result" is set to 1. If the user clicks on the "No" button, "result" is set to zero.

Example:

```
yesnobox 'Try agian?' 'Tera Term'  
if result goto retry  
end
```

## Appendixes

Error messages

About new-line characters

Note on negative integer constants

ASCII code table

## Error messages

Error message	Meaning
Can't call sub.	Cannot call the subroutine, the subroutine is located in a

Can't link macro.	different file. Failure to establish the link between MACRO and Tera Term.
Can't open file.	The include file does not exist, or there are too many nested include files.
")" expected.	A closing parenthesis does not exist where it should.
Link macro first.	The command cannot be executed before the link between MACRO and Tera Term is established.
Divide by zero.	The expression attempts to divide by zero.
Invalid control.	Invalid use of "else", "elseif" or "endif".
Label already defined.	Duplicate use of the label.
Label required.	The label is not defined.
Stack overflow.	There are too many nested subroutines, "for-next" loops or "while-endwhile" loops.
Syntax error.	The format of the statement is invalid.
Too many labels.	MACRO can not handle more than 256 labels.
Too many variables.	MACRO cannot handle more than 128 integer variables and 128 string variables.
Type mismatch.	The type of the constant or variable is invalid.
Variable not initialized.	The variable must be initialized before it is referenced.

## About new-line characters

New-line characters (CR or CR+LF) received from the host are converted to CR+LF pairs by Tera Term, and then Tera Term sends them to MACRO.

You should use the pair (CR+LF) as a new-line character to send to Tera Term.

ASCII code 13 (decimal) is for CR, and 10 is for LF.

Example:

**send 'abc'#13#10**

Same as the statement "sendln 'abc'". The actual new-line character to be sent to the host is determined by Tera Term.

**Wait #10'abc' 'def'#13**

Waits for a line beginning with "abc" or a line ending with "def".

**Logwrite 'abc'#13#10**

Writes line "abc" to the log file.

## Note on negative integer constants

Using a negative integer constant may cause a problem like the following:

For example,

**for i -10 0**

causes the syntax error, because the second parameter is regarded as "i-10" instead of "i". To avoid this problem, take one of the following solutions:

1) Put "0" before "-".

**for i 0-10 0**

2) Assign the negative constant to a variable.

**A = -10**

**for i A 0**

## ASCII code table

For example, the ASCII code for "A" is 65 in decimal or \$41 in hexadecimal.

Char	Code	Char	Code	Char	Code	Char	Code
NUL	0	DLE	16	SPACE	32	0	48
(^@)	\$00	(^P)	\$10		\$20		\$30

<b>SOH</b>	1	<b>DC1</b>	17	<b>!</b>	33	<b>1</b>	49
<b>(^A)</b>	\$01	<b>(^Q)</b>	\$11		\$21		\$31
<b>STX</b>	2	<b>DC2</b>	18	<b>"</b>	34	<b>2</b>	50
<b>(^B)</b>	\$02	<b>(^R)</b>	\$12		\$22		\$32
<b>ETX</b>	3	<b>DC3</b>	19	<b>#</b>	35	<b>3</b>	51
<b>(^C)</b>	\$03	<b>(^S)</b>	\$13		\$23		\$33
<b>EOT</b>	4	<b>DC4</b>	20	<b>\$</b>	36	<b>4</b>	52
<b>(^D)</b>	\$04	<b>(^T)</b>	\$14		\$24		\$34
<b>ENQ</b>	5	<b>NAK</b>	21	<b>%</b>	37	<b>5</b>	53
<b>(^E)</b>	\$05	<b>(^U)</b>	\$15		\$25		\$35
<b>ACK</b>	6	<b>SYN</b>	22	<b>&amp;</b>	38	<b>6</b>	54
<b>(^F)</b>	\$06	<b>(^V)</b>	\$16		\$26		\$36
<b>BEL</b>	7	<b>ETB</b>	23	<b>'</b>	39	<b>7</b>	55
<b>(^G)</b>	\$07	<b>(^W)</b>	\$17		\$27		\$37
<b>BS</b>	8	<b>CAN</b>	24	<b>(</b>	40	<b>8</b>	56
<b>(^H)</b>	\$08	<b>(^X)</b>	\$18		\$28		\$38
<b>HT</b>	9	<b>EM</b>	25	<b>)</b>	41	<b>9</b>	57
<b>(^I)</b>	\$09	<b>(^Y)</b>	\$19		\$29		\$39
<b>LF</b>	10	<b>SUB</b>	26	<b>*</b>	42	<b>:</b>	58
<b>(^J)</b>	\$0A	<b>(^Z)</b>	\$1A		\$2A		\$3A
<b>VT</b>	11	<b>ESC</b>	27	<b>+</b>	43	<b>;</b>	59
<b>(^K)</b>	\$0B	<b>(^I)</b>	\$1B		\$2B		\$3B
<b>FF</b>	12	<b>FS</b>	28	<b>,</b>	44	<b>&lt;</b>	60
<b>(^L)</b>	\$0C	<b>(^N)</b>	\$1C		\$2C		\$3C
<b>CR</b>	13	<b>GS</b>	29	<b>-</b>	45	<b>=</b>	61
<b>(^M)</b>	\$0D	<b>(^J)</b>	\$1D		\$2D		\$3D
<b>SO</b>	14	<b>RS</b>	30	<b>.</b>	46	<b>&gt;</b>	62
<b>(^N)</b>	\$0E	<b>(^^)</b>	\$1E		\$2E		\$3E
<b>SI</b>	15	<b>US</b>	31	<b>/</b>	47	<b>?</b>	63
<b>(^O)</b>	\$0F	<b>(^_)</b>	\$1F		\$2F		\$3F

<b>Char</b>	<b>Code</b>	<b>Char</b>	<b>Code</b>	<b>Char</b>	<b>Code</b>	<b>Char</b>	<b>Code</b>
<hr/>							
<b>@</b>	64	<b>P</b>	80	<b>`</b>	96	<b>p</b>	112
	\$40		\$50		\$60		\$70
<b>A</b>	65	<b>Q</b>	81	<b>a</b>	97	<b>q</b>	113
	\$41		\$51		\$61		\$71
<b>B</b>	66	<b>R</b>	82	<b>b</b>	98	<b>r</b>	114
	\$42		\$52		\$62		\$72
<b>C</b>	67	<b>S</b>	83	<b>c</b>	99	<b>s</b>	115
	\$43		\$53		\$63		\$73
<b>D</b>	68	<b>T</b>	84	<b>d</b>	100	<b>t</b>	116
	\$44		\$54		\$64		\$74
<b>E</b>	69	<b>U</b>	85	<b>e</b>	101	<b>u</b>	117

	\$45		\$55		\$65		\$75
<b>F</b>	70	<b>V</b>	86	<b>f</b>	102	<b>v</b>	118
	\$46		\$56		\$66		\$76
<b>G</b>	71	<b>W</b>	87	<b>g</b>	103	<b>w</b>	119
	\$47		\$57		\$67		\$77
<b>H</b>	72	<b>X</b>	88	<b>h</b>	104	<b>x</b>	120
	\$48		\$58		\$68		\$78
<b>I</b>	73	<b>Y</b>	89	<b>i</b>	105	<b>y</b>	121
	\$49		\$59		\$69		\$79
<b>J</b>	74	<b>Z</b>	90	<b>j</b>	106	<b>z</b>	122
	\$4A		\$5A		\$6A		\$7A
<b>K</b>	75	<b>[</b>	91	<b>k</b>	107	<b>{</b>	123
	\$4B		\$5B		\$6B		\$7B
<b>L</b>	76	<b>\</b>	92	<b>l</b>	108	<b> </b>	124
	\$4C		\$5C		\$6C		\$7C
<b>M</b>	77	<b>]</b>	93	<b>m</b>	109	<b>}</b>	125
	\$4D		\$5D		\$6D		\$7D
<b>N</b>	78	<b>^</b>	94	<b>n</b>	110	<b>~</b>	126
	\$4E		\$5E		\$6E		\$7E
<b>O</b>	79	<b>_</b>	95	<b>o</b>	111	<b>DEL</b>	127
	\$4F		\$5F		\$6F		\$7F