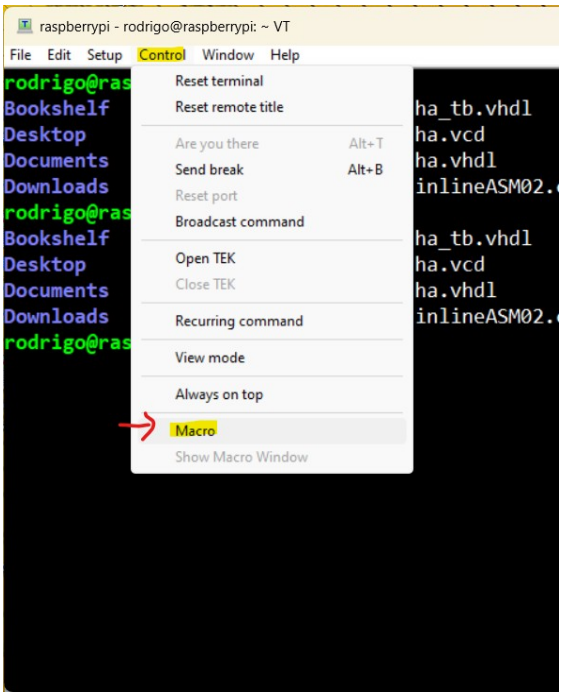
Tera Term Macros

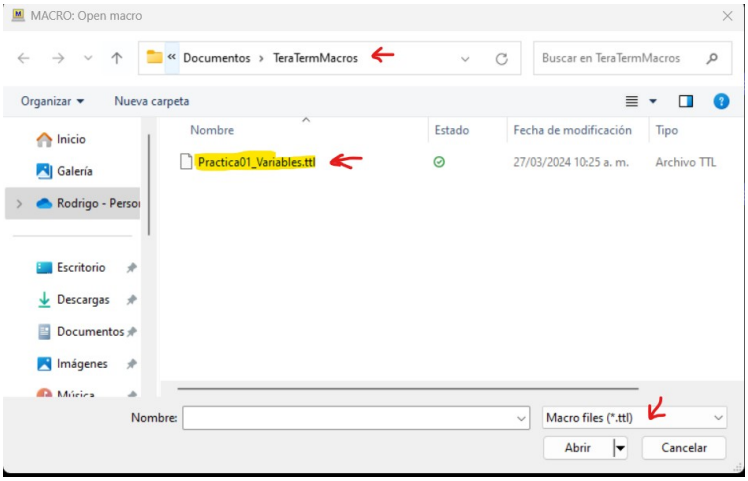
Introduccion a Tera Term

Ejecucion de una macro

Para ejecutar una macro desde la ventana de la consola de conexion, de la barra de menu, la opción *Control* y la opcion *Macro*

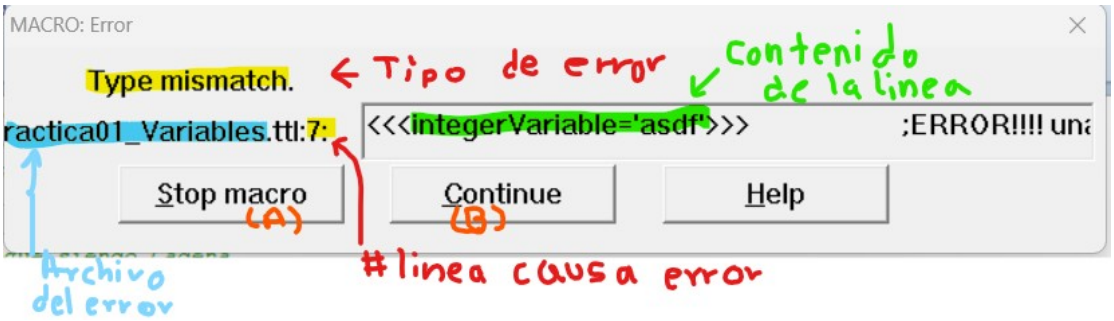


En el cuadro de Dialogo, navegue hasta el directorio que contiene la Macro que desea ejecutar:



Seleccione el archivo y de click en el boton de *Abrir*. La macro comenzara a ejecutarse y se detendra cuando encuentre un error, un cuadro de dialgo nos indicara la linea que produjo el

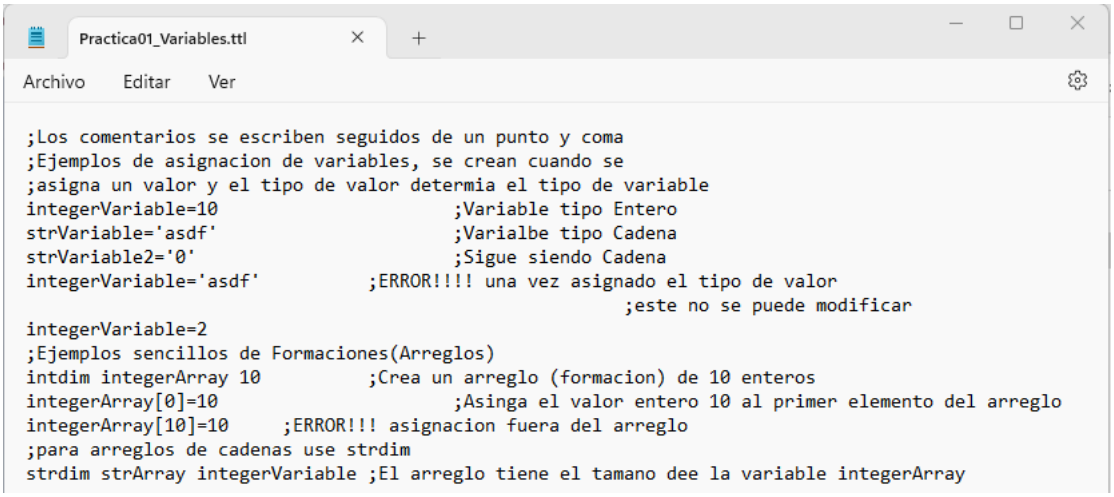
error y tendremos la opcion de detener la ejecución de la macro (A) o continuar con la ejecución(B).



En cierto sentido es muy parecido a la depuración de lenguajes interpretados, con la sutil diferencia de que el guión de instrucciones se copia en la ventana "expandiendo" las macros (lineas del guion), y a diferencia de la compilacion no se crea un archivo ejecutable.

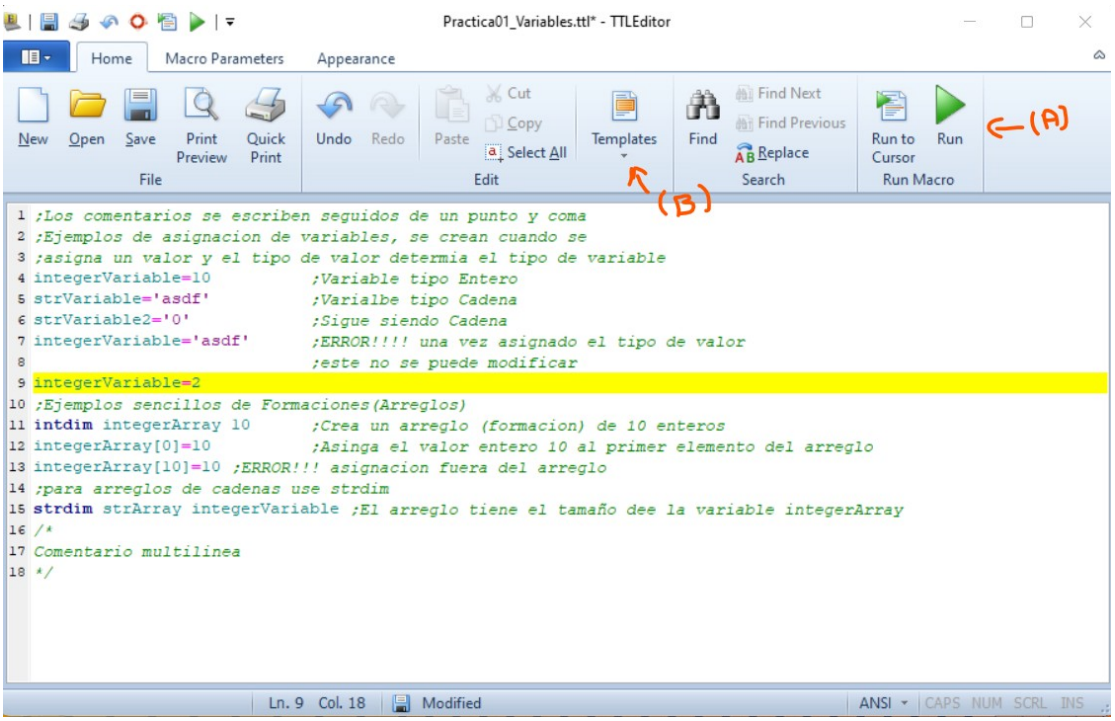
Edicion de Macros

El formato de un archivo de Tera Term macro es simplemente un texto plano con la terminacion.*ttl*. El editor de Block de Notas de Microsoft puede usarse para crear macros, tome el código del siguiente ejemplo:



Transcriba el contenido en el block de notas y guardelo con la extension *.ttl*, despues ejecutelo en la ventana de Tera Term como se mostro anteriormente.

Aparecera durante la ejecucion la ventana de Error por lo menos dos veces si selecciona la opcion del boton *continuar*, aunque esta macro no envia nada a la terminal, si realiza una inicialización de las variables, Usar al editor de block de notas para crear nuestras Macros puede ser util para macros de unas cuantas lineas, contar con un editor que ayude con la edicion hace la programación mas comoda, puede descargar un paquete para Vim o usar el editor de TeraTerm para macros en Windows:



El cual ofrece como adicional un boton rapido par ejecutar y probar nuestras macros(A), solo hay que enlazar nuestro archivo a la ventana de TeraTerm en la que deseamos ejecutar la macro, lo cual puede hacerse usando alguna de la plantillas (B) que el editor trae por defecto, entre las cuales existen plantillas para autologin comunes y muchas que son utiles cuando se llama al programa TeraTerm desde la linea de comandos (Asi es TeraTerm puede automatizarse dentro de un Script Batch de Windows o uno de Bash en Linux).

Descripcion del Lenguaje

Comentarios.

todo lo que siga despues del caracter **;** es un comentario, para los comentarios multilinea se pude usar:

/*

Todo entre estos caracteres es un comentario

*/

Tipos de Datos.

existen en el lenguaje de Macros de TeraTerm dos tipos de datos:

Enteros: IntegerVariable =10

Cadenas de Caracteres: strVariable='asdf' ;Con varios caracteres

strVariable='C' ;un solo carcter

y sus variantes en Arreglos (Formaciones), los cuales inician su indice en el valor 0

intdim IntArray 10 ;un Arreglo de Enteros de 10 elementos

strdim strArray intVariable; un Arreglo de Cadenas con una cantidad de elementos

;denotada por la variable intVariable

Sentencia de control IF

Sintaxis: if <int> <statement>

Ejecuta <statement>, So <int> no es cero

```
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
```

sentencia 'if' y 'elseif' deben terminar con 'then', 'elseif' y 'else' pueden omitirse 'endif' no puede ser omitido.

Ejemplos de codigo:

```
timeout=30      ;tiempo limite en segundos
wait 'value=0' 'ERROR' ;wesperar por uno de los dos valores
if result=1 then
;value =0 fue recibido de la terminal
elseif result=2
;ERROR fue recivido de la terminal
endif
```

```
timeout=1       ;Tiempo en segundos
mtimeout=500    ;Mas tiempo en milisegundos
wait 'Good' 'Bad'   ;esperara el tiempo 1.5 segundos
if resutl=0 then
;Ninguno de las dos opciones aparecieron en la
;terminal dentro de 1.5 segundos
endif
```

Enviando comando a la terminal (Send)

El comando Send envia caracteres hacia la terminal.

Sintaxis: send <'data1'> <data2> ...

Si <data> es una cadena, esta es enviada a la terminal. Si <data> es un entero, se envia el código del byte menor (0-255) codificado como ASCII.

Ejemplo: send 'ABC'

send 65 66 67 ; Es igual a enviar send 'ABC

myname='Tera Term'

send 'My name is ' myname ' . '

sendln variante que agrega al final de la cadena el caracter de nueva-linea(Enter)

sendfile sintaxis sendfile <filename> <binary flag> ;Tera term enviara el archivo <filename> hacia la terminal del host. Haciendo una pausa hasta terminar la trasferencia, si la casilla <binary flag> es no cero, el archivo es enviado sin ninguna modificación, si la casilla es cero, los caracteres de nueva linea son convertidos (CR-> CR/CRLF) y los caracteres de control excepto TAB, LF y CR son extraidos.

Ejemplo: sendfile 'data.dat' 1 ;enviar archivo 'data.dat' como esta.

Código de Ejemplo:

```
/* El script espera que la leyenda:
'root@localhost' aparezca para enviar
una instruccion a ejecutar en la terminal
mediante sendl 'Instruccion'
*/
cmdPrompt='root@localhost'
wait cmdPrompt
sendln 'Instruccion'    ;se agrega el ENTER
wait cmdPrompt
sendln 'Instruccion'
wait cmdPrompt
sendln 'Instruccion'
```

Loops (iteraciones)

Tenemos las opciones:

do while	do	for	until	while
loop	loop while	next	enduntil	endwhile

Consideracion especial para el uso del bucle *for*, cuando se deseea contar de manera invertida.

```
do while i > 0
   i = i-1
loop
;
i=10
do
  i=1-1
loop while i > 0
;          Para constantes negativas en loop For
for j 1 10  ;j 5 -1 es interpretado como 5-1
    i=j+i   ;use: j 5 0-1 o j 5 (-1)
next
;
i = 10
until  i > 0
    i=i-1
enduntil
;
i=10
while i > 0
    i=i-10
endwhile
```

Conviene revisar la documentacion de su version de Tera Term, es posible que algunas de las funciones Iterativas no esten presentes.

Sentencia Go to

Se recomienda el uso de Goto para manejo de errores, asi se evita terminar con codigo 'Spageti'.

Sintaxis: goto <label>

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

Ejemplo: goto label ;Jump to the next line of the ':label'.

...

:label

send 'abc'

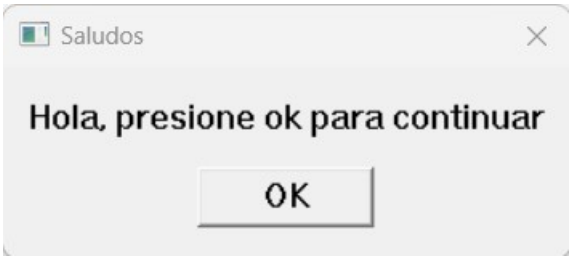
Código de ejemplo:

```
;No tiene tiempo de espera wait
wait 'Good' 'Bad'
if result=2 goto Error
;Mucho codigo iria aqui
exit ; El Programa termino correctamente
:Error  ;Etiqueta para manejar el Error
;Codigo para manejar el error
exit    ;Termino el programa con un error
```

Comandos Miscelaneos

messagebox Sintaxis: messagebox <message> <title>

Muestra un cuador de texto con <message> y <title>.



yesnobox Sintaxis: yesnobox <message> <title>

Muestra un cuador de texto con <message>, <title>, Boton "Yes"

y boton "No".

Si el usuario presiona el boton"Yes" , La variable de sistma "result" es ajustada a 1. En caso de presionar el boton "No", "result" es 0.

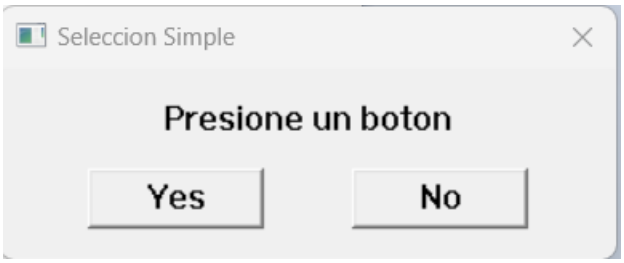
Ejemplo:

:retry

yesnobox 'Intentar nuevamente?' 'Tera Term'

if result goto retry

end



Código de Ejemplo:

messagebox ErrorMessage 'Error'

```
;Ejemplo de algunas ventanas basicas
messagebox 'Hola, presione ok para continuar' 'Saludos'
yesnobox 'Presione un boton' 'Seleccion Simple'
```

Anexo A

4. TTL command reference

Command index

New commands added for the current version are labeled by "** new **".

Commands modified for the current version are labeled by "** changed **".

4.1 Communication commands

4.1.1 bplusrecv

4.1.2 bplussend

4.1.3 changedir

4.1.4 closett ** changed **

4.1.5 connect ** changed **

4.1.6 disconnect ** new **

4.1.7 flushrecv ** new **

4.1.8 gettitle ** new **

4.1.9 kmtrecv

4.1.10 kmtsend

4.1.11 loadkeymap ** new **

4.1.12 logclose

.....

4.1 Communication commands

- -

4.1.1 bplusrecv

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.

- -

4.1.2 bplussend

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.

Example:

bplussend 'readme.txt'

- -

4.1.3 changedir

Format:

changedir <path>

Changes the current directory of Tera Term.

Example:

changedir 'c:\'

- -

4.1.4 closett ** changed **

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 TTPMACRO to it.

See also:

"4.1.5 connect"

"4.1.6 disconnect"

"4.1.29 unlink"

Example:

closett

connect 'host'

- -

4.1.5 connect ** changed **

Format:

connect <command line parameters>

If TTPMACRO is not linked to Tera Term, this command runs Tera Term

with <command line parameters>, and links it to TTPMACRO.

If TTPMACRO 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>.

If TTPMACRO has already been linked to Tera Term and Tera Term has

already been connected to the host, this command is ignored.

No other communication commands should be executed before the link is

established.

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

See also:

"4.1.4 closett"

"4.1.6 disconnect"

"4.1.29 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

- -

4.1.6 disconnect ** new **

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 TTPMACRO is kept.

See also:

"4.1.4 closett"

"4.1.5 connect"

"4.1.29 unlink"

- -

4.1.7 flushrecv ** new **

Format:

flushrecv

Clears received characters in the buffer of TTPMACRO.

Characters received from the host are transfered to TTPMACRO.

TTPMACRO stores the characters in the buffer and 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.

The "flushrecv" command can be used to avoid unexpected results of

character-reading commands caused by old characters in the buffer.

- -

4.1.8 gettitle ** new **

Format:

gettitle <strvar>

Retrieves the title text of Tera Term and stores it in the string

variable <strvar>.

Example:

gettitle titletext

- -

4.1.9 kmtrecv

Format:

kmtrecv

Causes Tera Term to receive a file from the host with the Kermit protocol.

Pauses until the end of the file transfer.

- -

4.1.10 kmtsend

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.

Example:

kmtsend 'readme.txt'

- -

4.1.11 loadkeymap ** new **

Format:

loadkeymap <filename>

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

Example:

loadkeymap 'keyboard.cnf'

- -

4.1.12 logclose

Format:

logclose

Causes Tera Term to close the log file.

- -

4.1.13 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
```

- -

4.1.14 logpause

Format:

```
logpause
```

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

- -

4.1.15 logstart

Format:

```
logstart
```

Causes Tera Term to restart the logging, if paused.

- -

4.1.16 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
```

- -

4.1.17 quickvanrecv

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.

- -

4.1.18 quickvansend

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.

Example:

```
quickvansend 'readme.txt'
```

- -

4.1.19 recvln ** new **

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:

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

See also "4.1.26 setsync" for the synchronous mode.

- -

4.1.20 restoresetup ** new **

Format:

restoresetup <filename>

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

Example:

restoresetup 'teraterm.ini'

- -

4.1.21 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 '.'

- -

4.1.22 sendbreak ** new **

Format:

sendbreak

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

- -

4.1.23 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

- -

4.1.24 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 (4.1.21).

Example:

sendln Only a new-line character is sent.

sendln 'abc'

Password='mypassword'

sendln Password

- -

4.1.25 setecho ** new **

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

- -

4.1.26 setsync ** new **

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 TTPMACRO.

TTPMACRO stores the characters in the buffer. The character-reading commands,

such as the "wait" command, read out the characters from the buffer.

Initially, TTPMACRO 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 transfering them to TTPMACRO. When the buffer regains enough space, Tera Term restarts receiving and transfering.

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 engough (4096 bytes) and all received characters are processed by character-reading commands before the buffer overflows.

See also "4.1.7 flushrecv" for clearing the buffer.

Example:

```
setsync 1       enter the synchronous mode
setsync 0       enter the asynchromous mode
```

- -

4.1.27 settitle ** new **

Format:

```
settitle <title>
```

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

Example:

```
settitle 'Tera Term'
```

- -

4.1.28 showtt ** changed **

Format:

```
showtt <show flag>
```

Minimizes Tera Term if <show flag> is zero.

Restores Tera Term if <show flag> is greater than zero.

Hides Tera Term if <show flag> is less than zero.

Example:

```
showtt 0        Minimize Tera Term.
```

showtt 1 Restore Tera Term.

showtt -1 Hide Tera Term.

- -

4.1.29 unlink ** new **

Format:

unlink

Terminates the link between the current Tera Term window and TTPMACRO.

TTPMACRO 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 TTPMACRO to it.

See also:

"4.1.4 closett"

"4.1.5 connect"

"4.1.6 disconnect"

Example:

connect 'host1' open a Tera Term window and link TTPMACRO to it

unlink terminate the link

connect 'host2' open another Tera Term window and link TTPMACRO to it

- -

4.1.30 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:

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

- -

4.1.31 waitevent ** new **

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 (opend).

The "waitevent" command returns the identifier of the actual event

in the system variable "result".

Example:

```
waitevent 4                     Wait the disconnection event
waitevent 2 or 8                Wait the unlink or connection events
```

if result=2 goto label1 The unlink event occured

if result=8 goto label2 The connection event occured

- -

4.1.32 waitln ** new **

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.

- -

4.1.33 waitrecv

Format:

waitrecv <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".

- -

4.1.34 xmodemrecv

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 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

- -

4.1.35 xmodemsend

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.

<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

- -

4.1.36 zmodemrecv

Format:

zmodemrecv

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

Pauses until the end of the file transfer.

- -

4.1.37 zmodemsend

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 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

.....

4.2 Control commands

- -

4.2.1 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.
```

- -

4.2.2 end

Format:

```
end
```

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

- -

4.2.3 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"
```

- -

4.2.4 exit

Format:

```
exit
```

Exits the include file and returns to the main file.

Example:

See "4.2.8 include".

- -

4.2.5 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              Repeat ten times.
  sendln 'abc'
next
for i 5 1               Repeat five times.
  sendln 'abc'
next
```

- -

4.2.6 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'
```

- -

4.2.7 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
```

- -

4.2.8 include

Format:

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

Moves control to the include file.

Example:

```
----- main file 'main.ttl' ------
i=10
:loop
include 'sub.ttl'                    Move to the include file.
if i>=0 goto loop
end
----- 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
----- End of 'sub.ttl' ----------    Go back to the main file.
```

- -

4.2.9 pause

Format:

```
pause <time>
```

Pauses for <time> seconds.

Example:

pause 10 Pause for 10 seconds.

pause Time

- -

4.2.10 return

Format:

return

Exits the subroutine and returns to the main routine.

Example:

See "4.2.1 call".

- -

4.2.11 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 Repeat ten times.

endwhile

.....

4.3 String operation commands

- -

4.3.1 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:

str2int val '123' val=123, result=1

str2int val '123abc' result=0

- -

4.3.2 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>	

Example:

strcompare 'abc' 'def' result = -1

strcompare command 'next'

if result=0 goto label

strcompare command 'end'

if result=0 end

- -

4.3.3 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'

- -

4.3.4 strcopy

Format:

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

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

The substring begings at the <pos>th character in <string>, and its length

is <len>.

Example:

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

- -

4.3.5 strlen

Format:

strlen <string>

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

Example:

strlen 'abc' result = 3

- -

4.3.6 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

.....

4.4 File operation commands

- -

4.4.1 fileclose

Format:

fileclose <file handle>

Closes the file specified by <file handle>.

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

Example:

fileclose fhandle

- -

4.4.2 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'
```

- -

4.4.3 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'
```

- -

4.4.4 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'
```

- -

4.4.5 filedelete

Format:

```
filedelete <filename>
```

Deletes the file specified by <filename>.

Example:

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

- -

4.4.6 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
```

- -

4.4.7 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:

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

- -

4.4.8 filerename

Format:

filerename <file1> <file2>

Renames <file1> to <file2>.

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

Example:

filerename 'test.dat' test2.dat'

- -

4.4.9 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'

- -

4.4.10 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 <offset> is 2.

Example:

fileseek fhandle 0 0 Move to the beginning of the file.

fileseek fhandle 10 1 Move 10 bytes from the current position.

fileseek fhandle 0 2 Move to the end of the file.

- -

4.4.11 filestrseek

Format:

filestrseek <file handle> <string>

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

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

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        Search for the string 'abc'
filestrseek fhandle 'abc'                in the file 'teraterm.log'.
if result=0 goto not_found
filereadln fhandle str                       Read characters from the next
                                             of the 'abc' to the end of the
                                             line.
:not_found
fileclose fhandle
```

- -

4.4.12 filewrite

Format:

```
filewrite <file handle> <string>
```

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

Example:

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

- -

4.4.13 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---------'
```

.....

4.5 Password commands

- -

4.5.1 delpassword ** new **

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 "4.5.2 getpassword" for the password file.

Example:

```
delpassword 'password.dat' 'mypassword'
```

- -

4.5.2 getpassword ** new **

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 sotred 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
```

- -

4.5.3 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'

.....

4.6 Miscellaneous commands

- -

4.6.1 beep

Format:

beep

Makes a beep sound.

- -

4.6.2 closesbox ** new **

Format:

closesbox

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

Example:

See "4.6.15 statusbox".

- -

4.6.3 exec

Format:

exec <command line>

Runs an application specified by <command line>.

Format:

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

- -

4.6.4 getdate

Format:

getdate <strvar>

Returns the current date in the string variable <strvar>, with the format

"YYYY-MM-DD".

Example:

getdate datestr

- -

4.6.5 getenv ** new **

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
```

- -

4.6.6 gettime

Format:

gettime <strvar>

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

Example:

```
gettime timestr
```

- -

4.6.7 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
```

- -

4.6.8 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".
```

- -

4.6.9 messagebox

Format:

messagebox <message> <title>

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

Example:

```
messagebox ErrorMessage 'Error'
```

- -

4.6.10 setdate ** new **

Format:

setdate <date>

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

Example:

```
setdate '1997-06-30'
```

- -

4.6.11 setdlgpos ** new **

Format:

setdlgpos <x> <y>

Changes the initial position for dialog boxes opend 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:

```
setdlgpos 0 0
messagebox 'Message' 'Title'  message box at the upper left corner
setdlgpos 0 200                    open the status box
statusbox 'Message' 'Title'
for i 0 200
  setdlgpos i 200                  moves the status box
next
```

- -

4.6.12 setenv ** new **

Format:

setenv <env name> <env value>

Sets the environment variable specified by <env name> to the character

string <env value>.

Example:

setenv 'WORK' 'c:\work'

- -

4.6.13 settime ** new **

Format:

settime <time>

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

Example:

settime '01:05:00'

- -

4.6.14 show ** changed **

Format:

show <show flag>

Minimizes TTPMACRO, if <show flag> is zero.

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

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

Example:

show 0	Minimize TTPMACRO.
show 1	Restore TTPMACRO.
show -1	Hide TTPMACRO.

- -

4.6.15 statusbox ** new **

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 (see 4.6.11) changes the position of status

dialog box.

The "closesbox" (see 4.6.2) command closes the status dialog box.

Example:

setdlgpos 200 200	set the initial position
statusbox 'Message' 'Title'	display the status dialog box

```
pause 3
setdlgpos 0 0                                   move the dialog box
pause 3
closesbox                                       close the dialog box
```

- -

4.6.16 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
```