Script Language Quick Reference

Script Reference

TERM has an extensive script language. Each command is explained in detail within the on-line help. This section is meant to provide a quick reference to the TERM script commands and functions and their parameters.

SCRIPT COMMANDS

ABORT	Aborts script file execution
	ABORT ABORT *retval*
ANSWER	Provides modem auto-answer control
	ANSWER [OFF, NOW, WAIT, or AUTO]
ATTR	Sets screen attributes for script file display.
	ATTR [DEFAULT, REVERSE, BLINK, UNDERLINE, BOLD, DIM]
AUTOCMD	Specifies the script to execute after connecting.
	AUTOCMD *filename*
BAUD - PARITY - STOPBITS - WORDLEN	Sets various communications line characteristics.
	BAUD *baudrate* [, *parity* [, *wordlen* [, *stopbits*]]] PARITY [NONE, ODD, EVEN, SPACE, or MARK] STOPBITS [1 or 2] WORDLEN [5, 6, 7, or 8]
BELL	Sounds beep.
	BELL
BREAK	Sends a break sequence.
	BREAK
CAPTURE	Sets options for data capture requests.
	CAPTURE NONE CAPTURE DISK *filename* [*options*] CAPTURE DEVICE *devicename* CAPTURE FILE *filename* [BINARY | MNEMONIC] CAPTURE SPOOL [*spoolet*] CAPTURE TERMINAL CAPTURE ON | OFF | CLOSE
CD	Changes current working directory or drive
	CD new_path
CDELAY, LDELAY, TDELAY	Sets time delays between transmitted characters, line feeds and tabs.

CDELAY *n* LDELAY *n* TDELAY *n*

SCRIPT COMMANDS

CHARSET	Specifies the National Replacement Character Set for use with VT220-7 and VT320-7 emulations.
	CHARSET [AMERICAN, BRITISH, DUTCH, FINNISH, FRENCH, CANADIAN, GERMAN, ITALIAN, DANISH, NORWEGIAN, PORTUGUESE, SPANISH, SWEDISH, or SWISS]
CLS	Clears display screen or active window.
	CLS
COLOR	Specifies script display screen colors for various attributes .
	COLOR [DEFAULT | REVERSE | BLINK | UNDERLINE] *fcol*, *bcol* COLOR [BOLD | DIM] *fcol*, *bcol* COLOR NUMBER *number*
COMMENT - REMARK	Comments script files.
	COMMENT anything REMARK anything * anything ! anything
CONFIG	Reads and writes Configuration Files.
	CONFIG NEW CONFIG GET [*session*] CONFIG SET CONFIG READ *filename* CONFIG WRITE *filename*
CONNECT	Establishes a connection using a preset configuration.
	CONNECT [config file]
COPY	Copies one file to another.
	COPY source destination
CREAD - CREADINT	Reads characters from the commline to a string or integer variable.
	CREAD variable [, timeout, nchars] [,option] CREADINT variable [, timeout, nchars]
CURSOR	Changes the shape of the cursor.
	CURSOR [ARROW, WAIT]

SCRIPT COMMANDS

DDE ADVISE	Requests an advise link on an item from the server application.
	DDE ADVISE *chan*, *item*, *var*
DDE CLOSE	Terminates a DDE conversation.
	DDE CLOSE *chan*
DDE [ENABLE | DISABLE]	Controls whether TinyTERM will accept DDE conversation requests.
	DDE [ENABLE, DISABLE]
DDE EXECUTE	Sends a command to a DDE server.
	DDE EXECUTE *chan*, *str*
DDE INIT	Initializes a DDE conversation with a DDE server.
	DDE INIT chan, service, topic
DDE NAME	Changes TinyTERM's DDE server name.
	DDE NAME *name*
DDE POKE	Sends data to a DDE server.
	DDE POKE *chan*, *item, str*
DDE REQUEST	Requests data from the server of an open DDE conversation.
	DDE REQUEST *chan*, *item, var*
DDE TIMEOUT	Sets the time TinyTERM will wait for a reply before timing out on a DDE conversation.
	DDE TIMEOUT *n*
DDE UNADVISE	Closes an advise link on an item on a DDE conversation.
	DDE UNADVISE *chan*, *item*
DEVPREFIX	Specifies the OpenNET device prefix.
	DEVPREFIX *str*
DIALOG	Executes a dialog string.
	DIALOG *string*
DIMSTR - DIMINT - DIMLOG	Creates string (DIMSTR), integer (DIMINT) or logical (DIMLOG) arrays.
	DIMSTR variable size DIMINT variable size DIMLOG variable size

SCRIPT COMMANDS

DISPLAY	Displays text or variable values on the terminal screen.
	DISPLAY *expr* [, *expr*] [;] ? *expr* [, *expr*] [;]
DLGOPEN - WOPEN	Creates dialog windows for data entry.
	DLGOPEN id, yloc, xloc, ylen, xlen [,fcol, bcol [,style [, title [,label]]]] WOPEN id, yloc, xloc, ylen, xlen [,fcol, bcol [,style [, title [,label]]]]
DO	Executes a file or procedure .
	DO filename [parameters] DO filename@proc [parameters] DO @proc
DTIMEOUT	Specifies the time to wait betweenredials.
	DTIMEOUT *secs*
DWAIT - WAIT	Waits for timeout or string match from communications line.
	DWAIT [CARRIER or NOCARRIER] DWAIT [*secs*] [, *timeout*] DWAIT [*match string*] [, *timeout*] WAIT [CARRIER or NOCARRIER] WAIT [*secs*] [, *timeout*] WAIT [*match string*] [, *timeout*]
EDIT	Runs a standard editor.
	EDIT [match string]
EDITOR	Specifies a standard editor.
	EDITOR *editor*
EMULATE	Selects the terminal type to emulate.
	EMULATE [VT320, VT320-7, VT220, VT220-7, VT100, VT102, VT52, ANSI, AT386, SCOANSI, WYSE60, WYSE60-25, PCTERM, TV950, WYSE50, TV925, TV912, ADM1, IB3101 or TTY]
ERASE	Erases a disk file.
	ERASE filename
EXECUTE	Inhibits script termination for one command.
	EXECUTE *str*

SCRIPT COMMANDS

EXMIT	Decrypts and XMIT the contents of a string variable.
	EXMIT *strvar*
EXTGET	Sets the external file transfer protocol string for GETting files.
	EXTGET *strvar*
EXTXFER	Sets the external file transfer protocol strings for XFERing files.
	EXTXFER *strvar*
FILLCHAR	Pre-fills buffer for XMODEM transfers.
	FILLCHAR *charnum*
FLAGS	Sets default TinyTERM startup flags.
	FLAGS flaglist
FLUSH	Discards characters received but not yet displayed.
	FLUSH
FONT	Sets the font used by TinyTERM for the emulation window.
	FONT NORMAL, fontname, charwidth, charheight
FOPEN - FCREATE - FREAD - FREADLN FWRITE - FWRITELN - FFLUSH - FSEEK - FCLOSE	Reads and write data files from scripts.
	FOPEN *n*, *filename*['R' or 'U'] FCREATE *n*, *filename* FREAD *n*, *variable*[,*bytecnt*] FREADLN *n*, *variable*[,*bytecnt*] FWRITE *n*, *variable*[,*bytecnt*] FWRITELN *n*, *variable*[,*bytecnt*] FFLUSH *n* FSEEK *n*, *offset* [,*type*] FCLOSE *n*
FOR - ENDFOR	Provides wildcard filename expansion and looping in script files
	FOR *variable* IN (*list*) DO ENDFOR
FTPHOST	Sets remote host, username and password used for FTP file transfers.
	FTPHOST host, user, pass

SCRIPT COMMANDS

GET - RCV - XFER - SEND	Sends and receive files.
	GET *remotefilelocalfile*[*options*] RCV *localfile*[*options*] XFER *localfileremotefile*[*options*] SEND *localfile*[*options*]
HANGUP	Hangs up phone and logs out from remote system.
	HANGUP
HELP	Provides online help.
	HELP ['*topic*']
HTIME	Sets the handshake timeout for file transfers.
	HTIME *nsecs*
IF - IFERROR - ELSE - ELSEIF - ENDIF	Provides conditional execution in script files.

	IF *condition*
	ELSEIF *condition*2
	ELSE
	ENDIF
	or
	IFERROR *command*
	ELSE
	ENDIF

IGNORE	Ignores characters on received or transmitted data.
	IGNORE [INPUT or OUTPUT]*ignore string*... IGNORE RESET
INISETFILE	Sets the file name for subsequent .INI operations.
	INISETFILE *filename*
INISETSTR	Writes values into Windows .INI files.
	INISETSTR section, key, value
INTER	Puts TinyTERM in interactive script mode.
	INTER
ITIME	Sets the idle time during ASCII file transfers.
	ITIME *nsecs*
LCHAR	Defines ASCII line send protocol character.
	LCHAR *charnum*

SCRIPT COMMANDS

LEARN	Starts and stops learn mode.
	LEARN START *filename* LEARN STOP
LOGFILE	Specifies a logfile filename for error logging.
	LOGFILE *filename*[OVERWRITE] LOGFILE CLOSE
LOG	Writes a string into the last opened log file.
	LOG string
LOGIN	Specifies an automatic login sequence.
	LOGIN *string*
LOGOUT	Specifies an automatic logout sequence.
	LOGOUT *string*
MEMLIST	Displays all memory variable names and values.
	MEMLIST [L] MEMLIST SYS
MENU ENTRY	Creates items for the pulldown list portion of the Main Menu.
	MENU *id* ENTRY *title* EXECUTE *command*
MENU OPEN	Creates a menu item on TinyTERM's Main Menu.
	MENU *id* OPEN *title*
MKDIR	Creates a new disk directory.
	MKDIR *newdir*
MODE	Selects terminal emulation communications modes.
	MODE [FULL, HALF, DUMP, MNEMONIC, or CONTROL]
MSGBOX	Opens a predefined message dialog window with an icon and text.
	MSGBOX text, title, type
NODE	Sets the phone number or network node name for connecting.
	NODE [number or nodename]
NOERROR	Inhibits script termination for one command.
	NOERROR *command*

SCRIPT COMMANDS

NETPORT - NETVTNAME - NETDIALOG	Configures MSNET/NetBIOS port number, virtual terminal name and handshake.
	NETPORT *n* NETVTNAME *string* NETDIALOG *string*
NETSTIME	Sets the NetBIOS/MSNET send timeout value.
	NETSTIME *n*
NETWORK	Sets the network type.
	NETWORK *str*
ON - ENDON	Defines event-catching procedures.
	ON KEY *key* ON ERROR ON ABORT ENDON
PAGES	Sets the number of available screen memory pages in an emulation.
	PAGES *n*
PASSWORD	Specifies the password for use with the ^P character in DIALOG and LOGIN strings.
	PASSWORD *str*
PASTE LINK	Opens a client DDE link from the LINK format in the Windows clipboard, and write the results to the active communications link.
	PASTE LINK
PASTE TEXT	Pastes the TEXT contents of the clipboard to the active communications line.
	PASTE TEXT
PAUSE	Pauses script file execution.
	PAUSE [*message*]
PORT	Specifies default communications line or network access method.
	PORT portlist

SCRIPT COMMANDS

PRINTER	Sets up output options for virtual printer requests.
	PRINTER NONE PRINTER DEVICE*dev* [*options*] PRINTER DISK *filename* [*options*] PRINTER SPOOL [*spooler*] PRINTER CAPTURE PRINTER CLOSE PRINTER FLUSH PRINTER ON | OFF
PROC - ENDPROC	Defines a procedure within a script file.
	PROC *procname* ENDPROC
PROTOCOL	Sets communications line protocol used during ASCII file transfers.
	PROTOCOL [XON, ETX, NONE] PROTOCOL [QUIET ON|OFF, DCD ON|OFF]
QUIT	Quits TinyTERM and returns to the operating system.
	QUIT exitval
READ - READN - READINT	Reads characters from the keyboard to a variable.
	READ [string,] variable READN [string,] variable READINT [string,] variable
REDIAL	Specifies the number of times to attempt to establish a connection.
	REDIAL *count*
REDRAW	Redraws the emulation screen.
	REDRAW
RELEASE	Deallocates and releases unused variables
	RELEASE *variable* [,*variable*] RELEASE *
REMOTE	Executes a TinyTERM command on a remote system.
	REMOTE *command*
RENAME	Renames a file.
	RENAME *oldname newname* [OVERWRITE]

SCRIPT COMMANDS

REPEAT	Provides looping control in script files.
	REPEAT UNTIL *expr*
RESTART	Restartable file transfer control.
	RESTART *n*
RETRIES	Limits the number of bad packets before an abort in file transfers.
	RETRIES *n*
RETURN	Returns from a script file or procedure.
	RETURN [*retval*]
RMDIR	Removes a directory.
	RMDIR *dir*
RTIME	Sets packet receive timeout during file transfers.
	RTIME *nsecs*
RUN - RUNX	Runs another program from within TinyTERM.
	RUN command line RUNX command line
SCREEN COPY	Copies the text selected on the emulation window to the Windows clipboard.
	SCREEN COPY
SCREEN NOSELECT	Removes the highlight on the emulation window and deselects the text that was highlighted.
	SCREEN NOSELECT
SCREEN PRINT	Prints the current contents of the emulation window to TinyTERM's print system.
	SCREEN PRINT
SCROLLBACK	Allocates the number of lines for the scrollback buffer.
	SCROLLBACK *n*
SESSION	Creates, accesses and disconnects multiple sessions.
	SESSION NEW [*service*] SESSION GOTO *n* SESSION NEXT SESSION DISCONNECT

SCRIPT COMMANDS

SET	Sets various TinyTERM options ON or OFF.
	SET ABORT [ON or OFF] SET ADDCR [ON or OFF] SET ADDEOF [ON or OFF] SET ADDLF [ON or OFF] SET ALTCHK [ON or OFF] SET ALTKEYS [ON or OFF] SET BLOCKMODE [ON or OFF] SET BSDEL [ON or OFF] SET COMPRESS [ON or OFF] SET CONTROL [ON or OFF] SET DDTR [ON or OFF] SET DESTBS [ON or OFF] SET DIMDIM [ON or OFF] SET DLGUNITS [ON or OFF] SET DTR [ON or OFF] SET ECHO [ON or OFF] SET EOFOPEN [ON or OFF] SET ERROR [ON or OFF] SET ESC8BIT [ON or OFF] SET ESCCTL [ON or OFF] SET EXITDISC [ON or OFF] SET EXITDTR [ON or OFF] SET FILECONV [ON or OFF] SET KERMECHO [ON or OFF] SET LOG [ON or OFF] SET MASKPAR [ON or OFF] SET NOSCROLL [ON or OFF] SET REVDIM [ON or OFF] SET SCALING [ON or OFF] SET TABEX [ON or OFF] SET TOUPPER [ON or OFF] SET RTS [ON or OFF] SET VIEW [ON or OFF] SET WRAP [ON or OFF] SET XFERACK [ON or OFF] SET XFERSTAT [ON or OFF]
SETCOLOR	Sets TinyTERM's internal color number.
	SETCOLOR colornum, newvalue
SETKEY	Changes the actions of any keyboard keys.
	SETKEY *key meaning* SETKEY [RESET | RESETX]

SCRIPT COMMANDS

SETSYM	Sets graphics characters
	SETSYM symbol#, newstrval [,1]
SETVAR	Assigns values to variables
	SETVAR variable expr
SERVER	Invokes server mode for WTERMCRC and ZMODEM file transfers from a script.
	SERVER
STARTSERVER	Restartable file transfer control.
	STARTSERVER *string*
STDFILE	Displays a standard file dialog allowing the user to select a file, its type, and path.
	STDFILE strvar, [filename [,filter [, default_extension [,title [,type]]]]]
STDSAVE	Displays a standard save file dialog allowing the user to select a directory and file name for a file to be saved. The Save Configuration As STDSAVE follows:
	STDSAVE strvar [,filename [,default_extension [,title]]]
STIME	Sets the packet send timeout during file transfers.
	STIME *nsecs*
STOPSERVER	Restartable file transfer control.
	STOPSERVER *string*
SWITCH - ENDSWITCH	Provides multiple conditional execution in script files
	SWITCH CASE *condition*1 DEFAULT ENDSWITCH
TERMINAL	Enters terminal emulation mode from a script file
	TERMINAL NOCARRIER TERMINAL [*match string*] [,*exitkey*]
TERMID	Sets the terminal response for DEC VT emulations.
	TERMID *str*
TIME	Estimates file transfer times.
	TIME filelist [options]

SCRIPT COMMANDS

TRANS	Translates characters on received or transmitted data
	TRANS [INPUT or OUTPUT]*source dest* TRANS DISPLAY*source dest* TRANS RESET
TYPE	Displays and paginates a file on the screen.
	TYPE *filename*[(b)]
USER	Specifies the username for use with ^U command character in DIALOG or LOGIN strings.
	USER *str*
VERSION	Shows the TinyTERM version number
	VERSION
WCLOSE	Closes pop-up windows from script files.
	WCLOSE *n*[,*n* [,*n*]]
WHIDE	Hides a displayed window.
	WHIDE *n*[, *n* [, *n*]]
WILDSIZE	Allocates memory for wildcard filename expansion
	WILDSIZE *n*
WMOVE	Moves a previously defined window.
	WMOVE *n*, y*, x, z*
WRU - WRUCHAR	Defines the who-are-you answerback string and character.
	WRU *string* WRUCHAR *charnum*
WSELECT	Selects pop-up windows from script files.
	WSELECT *n*
WSHOW	Shows a hidden window.
	WSHOW *n*[, *n*[, *n*]]
WTITLE	Changes the title of a previously defined window.
	WTITLE *n*, *str*
XMIT	Transmits a character string out the commline.
	XMIT string

SCRIPT COMMANDS

XPROT	Sets protocol for file transfer commands.
	XPROT [TERMCRC, WTERMCRC, XMODEMCRC, KERMIT, COMPSRVB, XMODEM, YMODEM, ZMODEM, FTPASCII, LINE or EXTERNAL]

SCRIPT FUNCTIONS

ASC()	Converts a character to its ASCII value.
	SETVAR *n* ASC(*str*)
ATOD()	Converts a date/time string to an integer value.
	SETVAR *n* ATOD(*date*)
CDATE()	Converts TIME() return value into a date string.
	SETVAR *n* CDATE(*time*)
CHNAME()	Converts a number to an ASCII representation.
	SETVAR *s* CHNAME(*n*, *type*)
CHR()	Converts an integer to a single ASCII character.
	SETVAR *c* CHR(*number*)
CHRDY()	Returns a character pressed at the keyboard.
	SETVAR *c* CHRDY()
CHRIN()	Waits for a keypress and returns it as a string.
	SETVAR *c* CHRIN( )
COMIN()	Waits for character to be received at communications line and returns it as a string.
	SETVAR *c* COMIN()
COMST()	Tests if a character has been received at the communications line.
	IF COMST() ENDIF
CTIME()	Converts TIME() return value to a time string.
	SETVAR *s* CTIME(*time*)
ENCRYPT()	Encrypts strings for use with EXMIT, ^U, and ^P.
	SETVAR *s* ENCRYPT(*str*)
EVAL()	Evaluates a command and return result code.
	SETVAR *n* EVAL(*cmd*)
EXISTS()	Tests for the existence of a file.
	IF EXISTS(*file*) ENDIF

SCRIPT FUNCTIONS

FCONCAT()	Builds a full pathname from path, filename and extension.
	SETVAR *s* FCONCAT(*path*, *file*, *ext*)
FEOF()	Tests for end-of-file on a file that has been FOPENed.
	IF FEOF(*fileno*) ENDIF
FEXT()	Extracts the file extension from a full pathname.
	SETVAR *s* FEXT(*filename*)
FHEAD()	Extracts the filename from a full pathname.
	SETVAR *s* FHEAD(*filename*)
FIELD()	Returns a field from a simple delimited string.
	SETVAR *s* FIELD(*str*, *fieldno*, *fchar*)
FILEDATE()	Returns the integer date of file.
	SETVAR *n* FILEDATE(*file*)
FILEMODE()	Returns integer permission bits of file.
	SETVAR *n* FILEMODE(*file*)
FILESIZE()	Returns the size of file.
	SETVAR *n* FILESIZE(*file*)
FNAME()	Extracts the filename+extension from a complete pathname.
	SETVAR *s* FNAME(*filename*)
FTELL()	Returns current position of FOPENed file
	SETVAR *n* FTELL(*file*)
GETENV()	Gets the value of an environment variable.
	SETVAR *s* GETENV(*var*)
GETHOMEDIR()	Gets the home directory of the user currently executing TinyTERM.
	SETVAR *s* GETHOMEDIR()
GETUSERDIR()	Returns the user's directory.
	SETVAR *s* GETUSERDIR()

SCRIPT FUNCTIONS

HASCOLOR()	Determines if the system running TinyTERM supports color.

	IF HASCOLOR()
	ELSE
	ENDIF

INIGETSTR()	Reads a value from a Windows .INI file.
	SETVAR *s* INIGETSTR(*section*, *key*, *defaultstr*)
ISCOPY()	Returns true if a screen area has been selected for copying to the Windows clipboard.
	SETVAR *b* iscopy()
ISDIR()	Tests if a filename is actually a directory.

	IF ISDIR(*file*)
	ELSE
	ENDIF

ISPASTE()	Returns true if there is data in the Windows clipboard of the selected format.
	SETVAR *b* ispaste(*format*)
KEYID()	Accesses the TinyTERM key id number for a key name.
	SETVAR *n* KEYID(*str*)
KEYIN	Accesses the TinyTERM key id number for a key press.
	SETVAR *n* KEYIN()
KEYNAME()	Accesses the TinyTERM key name associated with the key identification number.
	SETVAR *s* KEYNAME(*n*)
LEFT()	Returns leftmost portion of string.
	SETVAR *s* LEFT(*str*, *len*)
LEN()	Returns length of a string.
	SETVAR *n* LEN(*str*)
MDMSTAT()	Returns serial port status bits.
	SETVAR *n* MDMSTAT()
MIDSTR()	Returns a substring of *str*.
	SETVAR *s* MIDSTR(*str*, *pos, len*)

SCRIPT FUNCTIONS

OPSYS()	Returns operating system.
	SETVAR *n* OPSYS()
POS()	Returns position of string 1 within string 2.
	SETVAR *n* POS(*str1*,*str2*,*start*)
RIGHT()	Returns the rightmost *len* characters of *str*.
	SETVAR *s* RIGHT(*str*, *len*)
STR()	Converts an integer to a string.
	SETVAR *s* STR(*number*)
STRCONV()	Converts a string for display or edit in AT READ.
	SETVAR *s* STRCONV(*str*, *type*)
STRHEX()	Converts an integer to its related hexadecimal string.
	SETVAR *s* STRHEX(*n*, *len*)
SUM()	Returns the crc/chksum of a string.
	SETVAR *n* SUM(*str*, *type, startval*)
SYSVAR()	Reports the value of TinyTERM system variables.
	SETVAR *v* SYSVAR(*id*)
TERMBITS()	Returns value of TinyTERM internal settings.
	SETVAR *n* TERMBITS()
TERMKEY()	Accesses the key id number for keys assigned to TinyTERM functions.
	SETVAR *n* TERMKEY(*keynum*)
TIME()	Gets the system time.
	SETVAR *n* TIME()
TOLOWER()	Converts a string to all lower case.
	SETVAR *s* TOLOWER(*str*)
TOUPPER()	Converts a string to all upper case.
	SETVAR *s* TOUPPER(*str*)
TRIM()	Trims all spaces of the right hand side of a string.
	SETVAR *s* TRIM(*str*)

SCRIPT FUNCTIONS

UNIQ()	Creates a unique file name from a template.
	SETVAR *s* UNIQ(*str*)
VAL()	Returns integer value of string.
	SETVAR *n* VAL(*str*)

Data Types

TERM has three basic data types:

Type	Range
integer	-1,073,741,824 to 1,073,741,823
string	0 to 255 characters in length
logical	TRUE or FALSE

Related Topics

Arrays, Variable

Data Operators

Integer Data and Operators

The following operators work with integer data:

+	addition
-	subtraction
*	multiplication
/	integer division (greatest integer less than or equal to the number)
%, mod	remainder operator
!, not	bitwise compliment (not)
&&, and	bitwise and
|, or	bitwise or
^, xor	bitwise exclusive or
<<, shl	bitwise shift left... lowest bit set to 0
>>, shr	bitwise shift right... highest bit set to 0

Division by zero is not allowed (/,% operators) and results in an error condition.

String Data and Operators

Note: The character ASCII NUL may not be used as part of a string, as it will be mistaken by TERM for a string termination character.

+	concatenation
=, eq	equal to
!=, ne	not equal to
>, gt	greater than
<, lt	less than
>=, ge	greater than or equal to
<=, le	less than or equal to

Logical Data and Operators

Internal representation of TRUE and FALSE are -1 and 0. This is important, since TERM allows the AND, OR, XOR and NOToperators on logical values to produce integer results.

&&, and	logical and operation
|, or	logical or operation
^, xor	logical exclusive or operation
!, not	logical not operation
=, eq	equal to

!=, ne	not equal to
>, gt	greater than
<, lt	less than
>=, ge	greater than or equal to
<=, le	less than or equal to

Arrays or Indexed Variables

You may create arrays of integer, string, or logical data types. These are called indexed variables.

The commands for creating arrays are:

DIMINT creates integer arrays
DIMSTR creates string arrays
DIMLOG creates logical arrays

Your script accesses an element of an indexed variable by placing the number of the desired element within square brackets ([]) directly following the name of the variable.

NOTE: The index of an indexed variable begins at 0. So an array of 10 elements would have elements numbered from 0 to 9.

Here is an example script creating and using an indexed variable:

```
DIMSTR X 10          ! declare a 10 element string array called X
SETVAR X[0] "sdf"    ! set the first element to "sdf"
SETVAR X[2] X[1]     ! copy second element to third
SETVAR X[9] "abc"    ! set the last element to "abc"
```

Indexed variables may be used anywhere regular variables are allowed. The indexed variable must be the same type that the command requires. The AT GET command cannot use indexed variables.

Related Topics

Data Types, Variables

Rules for Expressions

TERM expressions are built using a syntax similar to many high level languages, such as C or Pascal. The result type of an expression is the same as the variables involved.

The following precedence chart defines the order of evaluation of an expression. Expressions are evaluated by precedence and elements of the same precedence are evaluated from left-to-right. (Operators which can be either a symbol or a descriptive verb are listed by verb rather than by symbol.)

```
NOT
*, / , MOD
+, -, SHL, SHR
EQ, NE, LT, LE, GT, GE
AND
OR, XOR
```

Script Command **AT**

Purpose

AT positions the cursor on the dialog window and provides full data entry support.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] *control*

Parameters

y	vertical placement of the AT control
x	horizontal placement of the AT control
n	id number for the control
var	variable used for control initialization and get values
control	is one of the following control types with listed parameters:
BOX	*h*, *w*, *type*[, *title*]
BUTTON	*str*, *retval*, [,*keyname*]
CHECKBOX	[*h*,*w* [,*flags* [,*title*]]]
EDIT	*h*,*w*,*flags* [,*maxlen*]
ELIST	*h*,*w*,*flags* [,*str* [, *maxlen*]] LIST *str* [, *str* [, *str*,*str*...]] | ARRAY *arrayname*
FONT	*h*,*w* [,*flags* [, *title* [, *linkid*]]]
ICON	
KEYIN	
LISTBOX	[*h*,*w* [,*flags* [, *title*]]] LIST *str* [, *str* [, *str*...]] | ARRAY *arrayname*
METER	*h*,*w*
PICTURE	
POPUP	[*h*,*w* [,*flags* [,*title*]]] LIST *str* [, *str* [, *str*...]] | ARRAY *arrayname*
RADIO	[*h*,*w* [,*flags* [,*title*]]]
TEXT	*h*,*w*, *flags*, *text*

Description

NOTE: all *x*,*y*,*h*,*w* parameters depend on the setting of SET DLGUNITS to determine whether character-based or dialog units are used in placement.

Do not use AT CLEAR in Windows. Each dialog is an independent unit/object with it's own value storage space. So, the AT statements may contain valid data after WSELECTing another dialog. DOS does not do it this way. The AT CLEAR is important in DOS because there is only 1 value storage space, shared by all dialogs.

AT statements in TERM for MS Windows are nor limited to 64 as in TERM for MS DOS.

Script Command **AT BOX**

Purpose

To display a group box dip, or bump at the x,y coordinates.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] BOX *h*, *w*, *type* [, *title*]

Parameters

h	Height of the display area.
w	Width of the display area.

type	
0 - 3	group box
4	horizontal dip: h = 1 (to separate items in dialogs)
5	vertical dip: w = 1
6	horizontal bump: h = 1
7	vertical bump: w = 1
title	the title of the groupbox. It is displayed in a gray banner at the top of the box. The box dimensions must include space for the title banner.

Description

Displays a group box dip, or bump at the x,y coordinates. This control is for visually grouping other controls.

Script Command **AT BUTTON**

Purpose

To display a pushbutton at x,y.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] BUTTON *title*, *retval*, [,*keyname*]

Parameters

ID *n*	Used for the ID in event callbacks, and as the base number of the picture associated with the icon. See Appendix F: Resources for more information on button IDs.
title	The text displayed on text only buttons.
retval	The value returned in _retval by the dialog when the button is pressed.
keyname	The keyname to associate with the button. Pressing keyname is the same as clicking on the button.

Description

Displays a pushbutton at x,y. If ID is in the range of 1 to 49 a predefined button with bitmap and text is displayed. If ID is in the range of 200 to 249 a predefined dynamically-sized bitmap button is displayed.

Users may add button IDs in the range of 50 to 99 for bitmap with text buttons, and 250 to 399 for dynamically-sized bitmap buttons. Adding a bitmap with text buttons requires an ID and 6 bitmaps. Adding dynamically-sized bitmap button requires an ID, and 4 bitmaps. See Appendix F: Resources for more information on adding bitmap buttons.

Button IDs in the range of 100 to 199 are user defined text only buttons.

Script Command **AT CHECKBOX**

Purpose

To display a checkbox control at x,y.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] CHECKBOX [*h*,*w* [,*flags* [,*title*]]]

Parameters

h Height of the display area. 0 = use auto calculated height for control.

w Width of the display area. 0 = use auto calculated length. The calculated length is usually longer than the actual text. Must include the width of the checkbox, about 12 dialog units.

flags

0 text to the left of the checkbox
1 text to the right of the checkbox

title The text to the checkbox.

Description

Displays a checkbox control at x,y. A checkbox returns 0 when it is not checked and 1 when it is checked.

Script Command **AT EDIT**

Purpose

To display an edit field at the x,y coordinates.**Script Usage**

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] EDIT *h*,*w*,*flags* [,*maxlen*]

Parameters

h Height of the display area. 0 = use auto calculated height for control.

w Width of the display area. 0 = use auto calculated length. The calculated length is usually longer than the actual text.

flags

1 allow only numeric input.
2 allow only hexadecimal input.
4 echo " " for each character input.
8 encrypt the input to the get var, echo " " for each character.

maxlen The maximum number of characters that can be entered into the field. Default is 240. The field automatically scrolls horizontally if the size of the field is shorter than *maxlen*. TERM strings are limited to 255 characters in length. If *maxlen* is longer than the display area, the input will scroll to allow *maxlen* characters to be entered.

Description

Displays an edit field at the x,y coordinates.

Script Command **AT ELIST**

Purpose

To display a list box with an edit field at the x,y coordinates.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] ELIST *h*,*w*,*flags* [, *maxlen*] LIST *str* [, *str* [, *str*...]] | ARRAY *arrayname*

Parameters

h Height of the display area. 0 = use auto calculated height for control.

w Width of the display area.0 = use auto calculated length. The calculated length is usually longer than the actual text.

flags

0 return selected string.
1 return index of selected string, first string is number 0.

maxlen The maximum number of characters to accept in to the input box. Defaults to 240 characters. The field automatically scrolls horizontally if the size of the field is shorter than *maxlen*. TERM strings are limited to 255 characters in length. If *maxlen* is longer than the display area, the input will scroll to allow *maxlen* characters to be entered.

Description

Displays a list box with an edit field at the x,y coordinates. Displays as an edit field with an arrow separate by a character space. This field may make selections from the list, or accept direct user input.

Script Command **AT FONT**

Purpose

To display a font list box at the x,y coordinates.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] FONT *h*,*w*,*flags*, *title* [, *linkid*]

Parameters

h Height of the display area. 0 = use auto calculated height for control.

w Width of the display area.0 = use auto calculated length. The calculated length is usually longer than the actual text.

flags

0 font name list box
1 font size listbox

title ignored

linkid used when *flags* = 0 to link the size list box with the font selected. It is the id of the font size list box.

Description

Displays a font list box at the x,y coordinates. There are two font boxes, one lists the name of the fonts, the other lists the sizes. The sizes list box is linked to the font name list box through *linkid*. When the size listbox is linked to the font name listbox, the sizes list is automatically updated to show the sizes available in the font selected in the font name list box.

See Also

See W_FONT.CMD for an example of this control.

Script Command **AT ICON**

Purpose

To display an icon of ID n on the dialog at x,y.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] ICON

Parameters

ID *n* Used as the base name of the picture associated with the icon. Icon IDs range from 400 to 499. See Appendix F: Resources for more information on icons.

Description

Displays an icon of ID n on the dialog at x,y. See Appendix F: Resources for more information on adding icons.

Script Command **AT KEYIN**

Purpose

To display a popup containing the TERM function keynames (F1 and above).

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] KEYIN

Description

Displays a popup containing the TERM function keynames (F1 and above). Selections may be made by typing the first letter of the keyname and then highlighting the desired key, or by browsing the list.

Script Command **AT LISTBOX**

Purpose

To display a listbox at the x,y coordinates.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] LISTBOX [*h*,*w* [,*flags*]] LIST *str* [, *str* [, *str*...]] | ARRAY *arrayname*

Parameters

h Height of the display area. 0 = use auto calculated height for control.

w Width of the display area. 0 = use auto calculated length.

flags

0 return selected string.
1 return index of selected string, first string is number 0.

Description

Displays a listbox at the x,y coordinates.

Script Command **AT METER**

Purpose

To display a meter box at x,y.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] METER *h*,*w*

Parameters

h Height of the display area. 0 = use auto calculated height for control.

w Width of the display area. 0 = use auto calculated length. The calculated length is usually longer than the actual text.

Description

Displays a meter box at x,y. A meter is a bar that is colored based on the progress of some action such as file transfer.

Script Command **AT PICTURE**

Purpose

To display a bitmap of ID n on the dialog at x,y.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] PICTURE

Parameters

ID *n*	Used as the base of the picture associated with the icon.Picture IDs range from 450 to 499. See Appendix F: Resources for more information on pictures.

Description

Displays a bitmap of ID n on the dialog at x,y. See Appendix F: Resources for more information on adding bitmaps.

Script Command **AT POPUP**

Purpose

To display a popup list box at the x,y coordinates.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] POPUP [*h*,*w* [,*flags*]] LIST *str* [, *str* [, *str*...]] | ARRAY *arrayname*

Parameters

h	Height of the display area. 0 = use auto calculated height for control.
w	Width of the display area. 0 = use auto calculated length.
flags	0 return selected string. 1 return index of selected string, first string is number 0.

Description

Displays a popup list box at the x,y coordinates. Displays as a an entry field with a down arrow on the right. The down arrow drops the list down for selection.

The LIST and ARRAY functions are documented in the TERM Reference Manual.

Script Command **AT RADIO**

Purpose

To display a radio button control at the x,y coordinates.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] RADIO [*h*,*w* [,*flags* [,*title*]]]

Parameters

ID *n*	Defines the radio button group. Must be the same for all buttons of the same radio button group.
h	Height of the display area. 0 = use auto calculated height for control.
w	Width of the display area. 0 = use auto calculated length. The calculated length is usually longer than the actual text. Must include the width of the checkbox, about 12 dialog units.
flags	0 text to the left of the radio button 1 text to the right of the radio button
title	The title text to the radio button.

Description

Displays a radio button control at the x,y coordinates. A group of radio buttons represent several options of which only 1 option may be selected. For each group of radio buttons, the ID must match for all radio buttons in the group.

The returned value is an INT representing the number of the selected radio button. First radio button defined is number 0. The buttons are numbered in order of definition in the script file that creates the dialog.

Script Command **AT TEXT**

Purpose

To display static text onto the dialog window at the x,y coordinates.

Script Usage

AT *y*,*x*, [ID *n*] [TABSTOP] [GET *var*] TEXT *h*, *w*, *flags*, *text*

Parameters

h	Height of the display area. 0 = use auto calculated height for control.
w	Width of the display area.0 = use auto calculated length. The calculated length is usually longer than the actual text.
flags	0 left justified 1 centered 2 right justified within the text display area.
text	the string to be displayed.

Description

Displays static text onto the dialog window at the x,y coordinates.

NOTE: The ampersand character (&) will cause the next letter to be underlined. To display an ampersand character two ampersands are required (&&).

The maximum length of TERM strings is 255 characters.

Script Command **AT CLEAR**

Purpose

To reset any earlier AT GET statements.

Script Usage

AT CLEAR

Description

Resets any earlier AT GET statements by clearing the value storage space. This command effectively clears any variables assigned to controls on the dialog.

Script Command **AT READ CALLBACK**

Purpose

To register a procedure as a callback procedure for a dialog.

Script Usage

AT READ CALLBACK *scriptfile@procname* [NOWAIT]

Parameters

scriptfile	the name of the script file where the callback proc is located. There is no guarantee that the script containing the callback will be active when the call back occurs, so the script file name should be included.
procname	the name of the proc to register as the callback procedure for the dialog.

NOWAIT — the keyword that allows the AT READ to exit without waiting for user input. Used when creating modeless dialogs or dialog ribbons.

Callback Procedure Parameters

The callback proc is called with the following string parameters:

_1 — Dialog window id number, as a string

_2 — message always 1004, as a string

_3 — control id affected, as a string

_4 — actions, as a string:
0 = mouse click,
1 = selection changed in list controls,
2 = mouse double click

Description

The Callback Proc is TERM script language's answer to event driven programming in an interpreted script language.

NOTE: The callback procedure should be placed in the top of a file to increase the speed of its execution.

The double click is received as 2 separate click events followed by a double click event. So don't do something on a double click you wouldn't want to do on a single click

Example

```
proc ribboncb
if _1 != "100"              ! if not the id of the factory ribbon bar return      return 0
endif
setvar wsave _wactive       ! save the id of the active window
wselect 0   ! select emulator screen for DISPLAY
at save val(_1)             ! read the values of the ribbon
switch
        case _3 = "201"             ! click on comm setup button
        do w_comm@open      ! execute the comm setup dialog
        case _3 = "207"             ! click on emulation setup button
        do w_emul@open      ! execute the emulation setup dialog
        ! process the other items of the ribbon by id
endswitch
wselect wsave               ! go back to the active window/dialog
return 1            ! inform windows to process the command also

endproc ribboncb
proc open
        DLGOPEN
        AT . . .
        AT . . .
```

```
        AT READ CALLBACK w_ribbon@ribboncb
endproc open
```

The full dialog ribbon callback procedure is located in W_RIBBON.CMD.

Script Command **CLS**

Purpose

Clear display screen or active window.

Script Usage

CLS

Description

This command clears the terminal screen.

CLS will clear the emulation window.

Related Topics

AT, DISPLAY

Script Command **COLOR**

Purpose

To specify script display screen colors for various attributes .

Script Usage

COLOR DEFAULT *fcol*, *bcol*
COLOR REVERSE *fcol*, *bcol*
COLOR BLINK *fcol*, *bcol*
COLOR UNDERLINE *fcol*, *bcol*
COLOR BOLD *fcol*, *bcol*
COLOR DIM *fcol*, *bcol*
COLOR NUMBER *number*

Parameters

fcol	Foreground color.
bcol	Background color.
number	Is the number of colors to use. Allowable values are 0, 2, 8, and 16. When *number* is set to zero, black and white are the only available colors.

Description

The COLOR command sets the foreground and background colors for the screen for a variety of situations. The COLOR DEFAULT command is used to set the screen color for normal terminal output. During terminal emulations, and when using the ATTR command, five other terminal attributes can be called for. With the COLOR command, these attributes can be assigned colors instead.

This command is ignored on monochrome systems or multiuser systems which do not have Sf and Sb in their termcaps. When you change the default color, you have to clear the screen to set the entire screen to that color. Otherwise, only the characters sent to the screen AFTER the color was changed will be in the new color. This also means that once a character is on the screen, it will stay the same color until erased.

For instance, on a color screen, it may be desirable to display white characters on a blue background, and red characters on a green background when reverse video is called for. This can be done with the COLOR DEFAULT and COLOR REVERSE commands, as shown in the examples below.

Allowable color names are:

BLACK	LTGREY
BLUE	LTBLUE
GREEN	LTGREEN
CYAN	LTCYAN
RED	LTRED
MAGENTA	LTMAGENTA
BROWN	YELLOW
WHITE	LTWHITE

COLOR NUMBER is used to change the number of supported colors.

For example, if your monitor can only display 8 colors, COLOR NUMBER 8 should be used to map high intensity colors to their low intensity equivalent.

For graphics mode, setting color to 8 will allow blinking characters. Setting color to 16 will allow 16 color display without blinking.

Setting *fcol* and *bcol* to DEFAULT will reset the color or attribute to its default value.

Examples

To make white on blue characters the normal display:

COLOR DEFAULT WHITE,BLUE

To make red on green characters for reverse video:

COLOR REVERSE RED,GREEN

Related Topics

Color Chart, ATTR, SETCOLOR

Script Commands **COMMENT - REMARK**

Purpose

To comment script files.

Script Usage

COMMENT *anything*

REMARK *anything*

anything

! *anything*

Parameters

anything	Is any one-line sequence of characters. It is ignored by TERM.

Description

The COMMENT command can be used to explain script files. It is useful then the same files are used by many users. The ! command is never echoed, even with SET ECHO ON. The ! comment may be used on the ends of lines. For example:

DISPLAY "State is "+state ! tell the state

The text "tell the state" will not be displayed.

Script Command **CONNECT**

Purpose

To establish a connection using a preset configuration.

Script Usage

CONNECT [*config file*]

Parameters

config file Is the name of a previously set up configuration (.TAP) file.

Description

The CONNECT command is used in a script file to establish a connection using the current PORT and NODE settings. With a *config file* specified, TERM establishes a connection using the settings defined in the specified file.

See connecting for a description of what TERM does when making a connection.

Typing CTRL_C while TERM is waiting for a carrier will abort the call.

NOTE You must hang up the phone (see the HANGUP command) before calling another number if a connection has already been established.

With a configuration file specified, TERM presets all configuration parameters from the configuration file and proceeds to dial a phone number (if specified in the file).

Many phone systems may require a delay between dialing digits. Most modems support a pause character; this is a comma (,) for Hayes compatible modems. For example, many systems require dialing a 9 to access an outside line and then pausing wait before dialing the rest of the number. To accomplish this:

NODE "9,277-8976"

CONNECT

If 2 seconds is not long enough, place more commas in the dial string.

Related Topics

Related topics: PORT, REDIAL, DTIMEOUT, DIALER, sysvar(), MODEM, LOGIN, LOGOUT, NODE

Script Command **REDIAL**

Purpose

To specify the number of times to attempt to establish a connection.

Script Usage

REDIAL *count*

Parameters

count Is the call redial count. If not specified, 3 redials will be attempted. Setting count to 0 will cause no redials.

Description

The REDIAL command is used to specify the number of times TERM will attempt to establish a connection.

Related Topics

CONNECT, DTIMEOUT, PORT, DIALER, HANGUP, MODEM, LOGIN, LOGOUT, NODE

Script Command **DTIMEOUT**

Purpose

To specify the time to wait between redials.

Script Usage

DTIMEOUT *secs*

Parameters

secs	Is the number of seconds to wait before redialing. If not specified, TERM will wait 60 seconds.

Related Topics

CONNECT, REDIAL, PORT, DIALER, HANGUP, MODEM, LOGIN, LOGOUT, NODE

Script Command **COPY**

Purpose

To copy one file to another.

Script Usage

COPY *source destination*

Parameters

source	Is the file to copy from. Wildcards are not allowed.
destination	Is the file to copy to. Wildcards are not allowed, and any existing file will be overwritten.

Description

This command is used to copy an entire file to another from within TERM's script language.

The *source* and *destination* file names are not quoted.

Examples

To copy the file one to the file two:

COPY one two

Related Topics

RENAME, ERASE, MKDIR, RMDIR, RUN, CD

Script Commands CREAD - CREADINT

Purpose

To read characters from the commline to a string (CREAD) or an integer (CREADINT) variable.

Script Usage

CREAD *variable* [, *timeout*, *nchars*] [,*option*]

CREADINT *variable* [, *timeout*, *nchars*]

Parameters

variable	Name of the variable to create and store the value in.
timeout	An optional timeout value in seconds. If 0 or no value is specified, it defaults to no timeout.
nchars	An optional character count. If 0 or no value is specified, it defaults to read until a carriage return is encountered. It ignores line feeds.
option	Runs CREAD characters through TRANS and IGNORE procedures. Controls which procedures to run characters through. 1 Check each character as it is read to see if it is defined as an IGNOREd character. If it is, do not store variable or decrement *nchars*. 2 Check each character as it is read to see if it is defined as a TRANSlated

character. If it is, store it as the translated value.

3 Check each character for both IGNORE and TRANSlate.

Description

This command reads characters from the communications line into a specified *variable*. By default, a line of text will be read in from the commline and used to create the named *variable*. CREAD creates a string-type variable, while CREADINT creates a numeric-type variable. The CREAD/CREADINT command terminates after either a timeout period has elapsed, *nchars* characters have been read, or a carriage return has been encountered. If timeout is not specified, no timeout will ever occur. For example, CREAD can use this format:

CREAD var,,3,1

A timeout must be specified. Therefore, if you do not want a timeout, use zero:

CREAD var,0,3,1

If nchars is not specified, then CREAD will terminate after a carriage return has been read. Characters are NOT echoed back to the person or machine doing the typing.

If nchars is not zero, all characters including carriage returns and line feeds will be included in the *variable*; otherwise only characters up to the first carriage return, ignoring all line feeds, will be included.

The CREAD command will decrement the character count when it encounters a NUL character, but will not store the NUL in the *variable*. This allows detection of received NUL characters, without prematurely terminating received string contents.

The CREAD command checks SET MASKPAR before setting the string variable. If MASKPAR is ON, CREAD will strip parity before setting the specified *variable*.

Examples

To read a line and put the results into the variable x:

CREAD x

To wait either for 10 characters to come in or for 3 seconds to elapse:

CREAD x,3,10

NOTE If only 5 characters were read in before the 3 seconds had passed, they will still be returned into x.

Related Topics

READ, SETVAR, MEMLIST, DIMSTR, IGNORE, TRANS, variables

Script Commands **DIMSTR - DIMINT - DIMLOG**

Purpose

To create string (DIMSTR), integer (DIMINT) or logical (DIMLOG) arrays.

Script Usage

DIMSTR *variable size*

DIMINT *variable size*

DIMLOG *variable size*

Parameters

variable	Is the name of the new array.
size	Is the size of the array.

Description

These commands are used to create arrays of TERM's basic data types: string, integer and logical values. Arrays must be declared before they are used.

A string array of length 10 declares 10 string variables, each capable of length 0-255 bytes. After declaration, each string is set to the null string, "". Integers are preset to 0, and logicals to false. Arrays are indexed variables indexed from 0, up to *size* -1.

Array variables may be fully deallocated using the RELEASE command, just like other variables.

The limit on the size of indexed variables is dependent upon the machine you are running on, and the amount of RAM it has. All variables reside in RAM.

Examples

To create a 10-element array x, and fill each element with a 1:

```
DIMINT x 10
SETVAR j 0
REPEAT
  SETVAR x[j] 1
  SETVAR j j+1
UNTIL j = 10
```

Related Topics

RELEASE, SETVAR, MEMLIST

Script Command **DISPLAY**

Purpose

To display text or variable values on the terminal screen.

Script Usage

DISPLAY *expr* [, *expr*] [;]

? *expr* [, *expr*] [;]

Parameters

expr	Is a string or numeric expression to be displayed on the screen.

Description

This command is used to display text or variables on the computer screen during a script file execution. It may be used to prompt the user to type something (such as a logon id), or remind him to do something. The message is displayed at the current cursor location. If a

semicolon is included after the expression, no new line is started. Otherwise, a new line is started.

Commas are used to separate variables specified with the command. This will allow variables of different types (for example, a numeric variable together with a string variable) to be displayed on the same line.

All DISPLAY output will be diverted through the emulation window.

A semicolon at the end of a DISPLAY statement will leave the cursor placed at the end of *expr*, otherwise, the cursor will be placed at the beginning of the next line.

Examples

If you are trying to display data on the last line of a window or the screen, use a semicolon:

```
DISPLAY "This is the last line";
```

The semicolon will keep TERM from scrolling the screen up after the information is displayed.

To display information immediately following other information, use a semicolon:

```
SETVAR rate 1200
? "The baud rate is ";
? rate
```

There is a space after is. As a result, when the message is printed, it looks like:

The baud rate is 1200

rather than

The baud rate is1200

DISPLAY is useful for debugging scripts with memory variables. The following shows one use:

```
SETVAR x "test string"
? "The length of variable x is: "+str(len(x))
```

Related Topics

AT, CLS, READ, PAUSE

Command **DO**

Purpose

To execute a script command file or procedure .

Script Usage

DO *filename* [*parameters*]

DO *filename*@*proc* [*parameters*]

DO @*proc*

Parameters

filename	Specifies the name of the script command file to execute.
proc	Is the name of a procedure to execute.
parameters	Are optional parameters that can be referenced within the DO script by the variables $1 through $9 and _1 through _9. Parameters are separated by spaces and may be enclosed in single or double quotes if needed.

Description

This command is used to execute a script command file or procedure. The file or procedure specified will be read by TERM and executed. Each line of the script file (terminated by a CR/LF or LF) must contain one valid command. TERM will produce an error if it doesn't recognize a command. The SET ECHO ON command can be used to see which commands are executed in the script file.

Before execution of each command in the script file, all $ variables are macro-substituted in the command line. This allows commands and parameters to be specified by variables. The SET ECHO ON statement will display the command line after variable substitution. See Using Variables.

After the script file has finished executing, TERM will return to the mode it was in when the DO command was given. DO commands can be given from within other script files so that script files can be nested.

Any ON/ENDON statements within a script file are only effective if TERM is presently executing within the script file that triggered the ON statement.

Examples

To execute a script file named univ.cmd,use

DO univ

To call a procedure named PROC1 in the script univ.cmd:

DO univ@proc1

To call a procedure named ONE within the current script:

DO @one

Related Topics

PROC, RETURN, SET, ON, MODEM, Parameters

Script Commands **DWAIT - WAIT**

Purpose

To wait for timeout or string match from communications line.

Script Usage

DWAIT [CARRIER or NOCARRIER]
DWAIT [*secs*] [, *timeout*]
DWAIT [*match string*] [, *timeout*]

WAIT [CARRIER or NOCARRIER]
WAIT [*secs*] [, *timeout*]
WAIT [*match string*] [, *timeout*]

Parameters

secs	The number of seconds to delay before executing the next command.
timeout	An optional number of seconds of line quiet to wait before terminating the command. If not specified, the command will never timeout.
match string	The character string that must be received before executing the next command. If the string begins with a number, quotes must be used to distinguish it from a *secs* parameter. Strings with special characters in them can be specified using escape characters. See Specifying Strings.

Description

These commands are used to delay execution of a script file for a number of seconds until a sequence of characters has been received, or until a datalink carrier has either just been

established or just been terminated. A CTRL+C (^C) typed during the wait will abort this command. The DWAIT command is used to capture data while waiting for a line timeout or a string to be received. If a capture file is on, received data will be captured as well as displayed on the screen.

The WAIT and DWAIT commands can take either a string variable or a string literal as their first argument. The string value allows the use of the "|" character to indicate a desire to wait for multiple strings at the same time. The _retval system variable is set to the values 0...n-1 respectively for which case was matched. Up to 10 strings may be simultaneously waited for.

For example, the command

DWAIT "RING|ON LINE"

waits for either the word RING or the two words ON LINE to be received. The system variable _retval will be set to either 0 or 1 on return from this command.

The WAIT command works like DWAIT except it never captures data to a capture file. If a *match string* is specified with WAIT, any characters received will be discarded up to and including the characters to be matched.

If SET MASKPAR OFF is set, TERM checks the incoming characters for 8-bit matches against the *match string*. If SET MASKPAR ON is set, TERM strips off the eighth bit of received data before comparing for a string match. TERM always discards received NULs (hex 00) before comparing.

For programming purposes, the global logical variable _timeout is set TRUE when a DWAIT or WAIT exceeds the *timeout* before a string match was found.

The NOCARRIER and CARRIER options are implemented under DOS only.

WAIT and DWAIT processing is independent of any conversions requested by the SET ADDCR, SET ADDLF, TRANS or IGNORE commands. In addition, null characters are always ignored. If SET MASKPAR ON is selected, TERM strips the high bit from any characters that are transferred.

Examples

To wait up to 10 seconds for a carriage return followed by the string login: to be received before continuing:

```
DWAIT ":",10
IF _timeout
  DISPLAY "Never said login:"
  RETURN
ENDIF
```

Related Topics

XMIT, CAPTURE, SET ADDCR, SET ADDLF, IGNORE, TRANS, ON

Script Command **EDIT**

Purpose

To run a standard editor.

Script Usage

EDIT [*match string*]

Parameters

match string Is the file name you want to edit.

Description

The EDIT command runs a standard editor on a file for use within script commands. For ease of interactive use, the last *match string* used is remembered if a *match string* is not entered.

If the EDIT command is specified and no previous EDITOR command was executed, TERM first looks for the environment variable EDITOR. If it is not found, then TERM attempts the standard editor for operating system. See EDITOR for a listing.

Examples

To edit the file foo:

EDIT foo

To edit the file foo again:

EDIT

Related Topics

EDITOR, RUN

Script Command **EDITOR**

Purpose

To specify a standard editor.

Script Usage

EDITOR *editor*

Parameters

editor Is the full path name of an editor to run with the EDIT command. This must be a text string in quote marks.

Description

The EDITOR command specifies the name of an editor to be used by the EDIT command. This command is normally set in the TERM.SYS file.

Default editors for various systems are:

DOS EDLIN
Windows NOTEPAD
UNIX vi

Examples

To specify Wordstar as the standard editor:

EDITOR "ws"

Related Topics

EDIT, RUN

Script Command **EMULATE**

Purpose

To select the terminal type to emulate.

Script Usage

EMULATE [VT320, VT320-7, VT220, VT220-7, VT100, VT102, VT52, ANSI, AT386, SCOANSI, WYSE60, WYSE60-25, PCTERM, TV950, WYSE50, TV925, TV912, ADM1, IB3101 or TTY]

Description

This command controls the video terminal emulated during terminal emulation. Different terminal types are selected depending upon the type of remote system the user is communicating with. The terminal types are discussed next:

ADM1 ADM 1 terminal emulation.

ANSI ANSI 3.64. Generic ANSI emulation.

AT386 AT&T UNIX System V/386 Color Console. This is a full-screen, 25 line emulation of the console on AT&T and Interactive UNIX System V/386 systems.

IB3101IBM 3101 terminal. Supports block and line mode.

PCTERMWyse WY-60 scancode mode emulation. Automatically selected if Wyse60 is selected and the application requests it.

SCOANSI SCO UNIX Color Console. This is a full-screen, 25 line emulation of the console on SCO UNIX systems.

TTY Pass-through mode. All characters received are displayed on the screen and no special escape processing is performed.

TV912 TeleVideo 912 terminal emulation.

TV925 TeleVideo 925 terminal emulation.

TV950 TeleVideo 950 terminal emulation.

VT320 DEC VT320 terminal emulation. Includes support for Multinational Character Sets, compose characters, label line, transparent printing, double-high/double-wide characters and numeric keypad.

VT320-7 DEC VT320 terminal emulation with National Replacement Character Set support which allows foreign character display over 7-bit connections.

VT220 DEC VT220 terminal emulation with Multinational Character Set support.

VT220-7 DEC VT220 terminal emulation with National Replacement Character Set support.

VT100 DEC VT100 terminal emulation.

VT52 DEC VT52 terminal emulation.

WYSE60Wyse WY-60 terminal emulation. Includes support for 132 columns, label line and 25-line mode.

WYSE60-25 Wyse WY-60 terminal emulation 25-line mode. Automatically selects 25-line mode.

WYSE50Wyse WY-50 terminal emulation.

Related Topics

MODE, TERMINAL, SETKEY

Script Command **ERASE**

Purpose

To erase a disk file.

Script Usage

ERASE *filename*

Parameters

filename Indicates the file name to erase. Wildcard characters are not allowed.

Description

This command may be used to erase files without leaving TERM and going to the operating system.

If file name cannot be erased, an error message is displayed.

Examples

To erase the file foobar.txt:

ERASE foobar.txt

Related Topics

RUN, COPY, RENAME, MKDIR, RMDIR, CD, IFERROR, NOERROR

Script Command **EXMIT**

Purpose

To decrypt and XMIT the contents of a string variable.

Script Usage

EXMIT *strvar*

Parameters

strvar a string which has been encrypted through the encrypt() function, or the AT...GET...ENCRYPT command.

Description

Equivalent to XMIT, but decrypts *strvar* before sending the string down the connection.

Related Topics

XMIT, encrypt()

Script Command **EXTGET**

Purpose

To set the external file transfer protocol string for GETting files.

Script Usage

EXTGET *strvar*

Parameters

strvar a string containing the name of the program to run. It must also include any arguments that the external protocol needs.

See your external protocol documentation for required arguments and syntax.

Description

EXTGET is usually used in TERM.SYS to set the string for external protocol support.

TERM's external protocol support consists of loading a shell and passing the shell the contents of the external protocol string. This allows use of DOS executable files and batch files as external protocol drivers. TERM uses the value of the COMSPEC environment variable to determine the shell used to run the external program. If not set, TERM uses COMMAND.COM.

TERM automatically adds three arguments to the external protocol string:

* the current port,
* the src string from the file transfer dialog,
* the dst string from the file transfer dialog.

If the command ends in .CMD. TERM executes a TERM script file as the external protocol driver. TERM does not load a system shell for script file external protocols. TERM will pass only source and destination information to the script file.

Examples

To execute an external Kermit protocol in receive mode:

EXTGET "KERMIT -r"

Related Topics

EXTXFER, File Transfers

Script Command **EXTXFER**

Purpose

To set the external file transfer protocol strings for XFERing files.

Script Usage

EXTXFER *strvar*

Parameters

strvar a string containing the name of the program to run. It must also include any arguments that the external protocol needs. See your external protocol documentation for required arguments and syntax.

Description

EXTXFER is usually used in TERM.SYS to set the string for external protocol support.

See EXTGET for the description of TERM's external file transfer support.

Examples

To execute an external Kermit protocol in send mode:

EXTXFER "KERMIT -s"

Related Topics

EXTGET, File Transfers

Script Command **FILLCHAR**

Purpose

To pre-fill buffer for XMODEM transfers.

Script Usage

FILLCHAR *charnum*

Parameters

charnum is the ASCII value of the fill character used to pad out TERM's internal 128-byte buffer used in the SEND/XFER command. TERM zero-fills this buffer by default. This may be any integer between 0 - 255.

Description

This command is typically used when transferring text files from UNIX to CP/M systems. This command is only valid when using XMODEM protocol. When using the SEND and XFER commands, TERM rounds up the number of bytes sent to the next 128-byte multiple, since there is no packet-length field in XMODEM protocol. This means that only whole packets can be sent, so some type of padding has to be used.

Examples

To gain full compatibility when transferring text files from UNIX to CP/M:

FILLCHAR 26

Related Topics

SET ADDEOF, toupper(), XPROT, File Transfer, XPROT

Script Command **FLAGS**

Purpose

To set default TERM startup flags.

Script Usage

FLAGS *flaglist*

Parameters

flaglist is a quoted string containing the default options TERM should use if none are specified on the command line.

Description

This command is used in the TERMSYS file to set default TERM startup options. If any options are given when TERM is executed, *flaglist* will be ignored.

The flaglist option should look exactly like options do on the command line.

The following table lists the option flags allowed by TERM.

Flags:

-c	Set Control mode for TERMCRC server.
-e	Set ECHO ON and VIEW ON for the debugging of scripts.
-h	Set Half Duplex terminal mode.
-lport:node	Same as the PORT:NODE commands.
-psize	Set the size of the print buffer in 1K increments. The default size is 512 bytes.
-sbaudstr	Set the baud rate and related parameters according to baudstr. Whenever you set the baud rate specification for TERM with an option flag, you should be aware that TERM is simultaneously setting the baud rate, parity, word length, and stop bits. You may also set these values independently. The available values for these parameters are as follows: Baud rate: 110, **3**00, **6**00, **12**00, **24**00, **48**00, **9**600, **144**00, **192**00, **384**00, **576**00 Items in **bold** represent default values. Parity: E (even), O (odd), **N (none)**, M (mark) or S (space)/line Word Length: 5, 6, 7 or **8** Stop bits: **1** or 2 Baud rates can be abbreviated to the shortest unique segment. For example, the common baud rate values of 9600, 1200, and 300 can be abbreviated to 9, 12, and 3 respectively, while 38400 could be abbreviated to 384. No matter what baud rate you select, parity is set by default to None, word length is set to 8, and stop bits are set to 1 unless otherwise specified. If you want to change these values, you need to enter them here with the baud rate. Separate the entries with commas. For example, to run TERM at 1200 baud with even parity and 7-bit data transmission, you would enter at your system prompt: WTERM -s12,e,7
-u	Omit startup banner and menu.
-v	Set VT100 terminal emulator.
-x	TERMCRC Server Mode. Use the current terminal (standard input and output) as the communications line. This option sets both -q and -c. TERM will respond to commands from a remote system running TERM until a CTRL+C is typed, after which the standard input and output will return to the shell. TERM sends the string OK when initialized in this mode.
-1x	WTERMCRC server mode.
-2x	ZMODEM server mode.
-z	Don't open any communications port at startup.

Examples

To set default startup options -z and -u:

FLAGS "-zu"

Related Topics

How to Change Startup Options

Script Command **FLUSH**

Purpose

To discard characters received but not yet displayed.

Script Usage

FLUSH

Description

This command discards any characters received by TERM, but not yet displayed or processed. This is useful when a high input baud rate and slow processing time allow the system to get behind on character display, and display output is no longer desired.

This command is also useful when TERM is reentered after a RUN command, and many characters have been received but not yet displayed.

In script files, you may want to use FLUSH to guard against noisy lines. For example, if you were about to send a file to a VMS system without transfer capabilities, you may have had line-noise that looked like a linefeed (or a linefeed may have been left over from the last command you did).

Flushing will prevent you from getting an off-by-one error:

```
XPROT ASCII
FLUSH
XMIT "COPY TT: FILE.TXT;1"
WAIT 1
FLUSH
SEND foo.dat
XMIT "^Z"
```

Related Topics

CREAD, DWAIT

Script Commands for data files

Purpose

To read and write data files from scripts.

Script Usage

FOPEN *n*, *filename*[,'R' or 'U']
FCREATE *n*, *filename*

FREAD *n*, *variable*[,*bytecnt*]
FREADLN *n*, *variable*[,*bytecnt*]
FWRITE *n*, *variable*[,*bytecnt*]
FWRITELN *n*, *variable*[,*bytecnt*]

FFLUSH *n*
FSEEK *n*, *offset*[,*type*]
FCLOSE *n*

SETVAR *offset* FTELL(*n*)
SETVAR bool FEOF(*n*)

Parameters

n	Is the file number, from 1 to 20.
filename	Specifies the path name of the file to use when opening or creating a file. The *filename* must either be a string variable or be enclosed in quotes.
variable	Is a string variable from/to which to read or write data. A quoted string may be used for FWRITE statements.
offset	Is the *offset* from the beginning of the file in bytes from which to start the next read or write. If the optional numeric parameter *type* is 1, the offset is relative to the current position; if 2, the offset is relative to the end of the file. The default (0) is relative to the beginning of the file.

Description

These commands allow both ASCII and binary data files to be read, written, and updated from within TERM. All files must be FOPENed or FCREATEd before they can be read from or written to. The FOPEN command is passed a file number, n, which may be from 1 to 20. All subsequent accesses use this number for reading and writing.

To open a file that previously exists, the FOPEN command is used. To create a new file or overwrite an existing one, the FCREATE command is used. The FOPEN command has two options that are used to either open an existing file for reading and writing (updating), or to open a file with only operating system READ permissions. The FCREATE command always opens for reading and/or writing, so these options are not needed. Thus, the third parameter to FOPEN is:

"R" Open for read only (default)
"U" Open for reading and writing

After a file has been opened, it can be read from or written to. The FREAD and FWRITE commands are used for both ASCII and binary data. If the optional third argument *bytecnt* is specified, then *bytecnt* binary bytes be either read or written. Otherwise, the FREAD command will read up to and including a linefeed from the file into the *variable*.

Since different operating systems store ASCII text file data using different end-of-line formats (i.e. UNIX uses linefeeds only, DOS and VMS use carriage return and linefeeds), TERM has the FREADLN and FWRITELN commands. These commands work the same as FREAD and FWRITE for binary files, but operate on ASCII files, thus releasing the programmer from having to know which text file format he is dealing with. The FREAD command reads up a full line of ASCII data from the file, discarding the carriage return and/or linefeed. The FWRITE command writes a line to an ASCII file, followed by the line delimiter appropriate for the operating system it is on.

If a file is FOPENed for updating, the FSEEK command must be used between reads followed by writes to the file. The FSEEK command sets the position of the next file read or write. The optional numeric third parameter, *type*, sets the offset types:

0 Relative beginning of file (default)
1 Relative current position
2 Relative end of file

After data has been written to a file, the file must be FCLOSEd to update the directory entry and flush internal buffers. This is automatically done on TERM exit, but should be executed as soon you are finished with the file to release internal memory buffers.

The ftell(n) function returns the current position in the file n, in bytes, relative to the beginning of the file.

The feof(n) function returns true if the current position is at the end of the file.

Examples

To create a data file FLIST with a line that reads "This is the first line of flist":

```
FCREATE 1,"flist"
FWRITELN 1,"This is the first line of flist"
FCLOSE 1
```

Related Topics

LOGFILE, CAPTURE, PRINTER, ERASE, COPY, SET LOG, uniq(), exists()

Script Commands **FOR - ENDFOR**

Purpose

To provide wildcard file name expansion and looping in script files

Script Usage

FOR *variable* IN (*list*) DO

.

(commands between FOR and ENDFOR are executed with the string *variable* set to each value in *list*)

.

ENDFOR

or

FOR *variable* IN (*list*) DO *command*

Parameters

list	Is a space-separated list of characters or file names. If *list* contains file name wildcards like asterisk (*), question mark (?), or brackets (), the *list* will be expanded to include all file names in the directory that match that specification. File names should NOT be quoted.
command	Is any valid TERM script command. This form of the FOR command is used interactively and does not include an ENDFOR statement.

Description

This command allows for setting a string *variable* and looping through a section of script commands. The FOR command can loop with a list of specified strings, search a directory for matching file names, or both. This command is typically used to search a directory for all files matching a specification, and then to perform some action on them.

The strings or file name specifications in list should not be quoted, as they are expanded verbatim.

The short form of the FOR command allows a script command immediately following the DO. An ENDFOR statement must not be given in this case. This form is most useful for interactive use or quick setting of array elements.

The FOR command may not be nested within another FOR command.

Examples

To search a directory for all files ending in .cmd:

```
FOR i IN (*.cmd) DO
   DISPLAY i
ENDFOR
```

or

```
FOR i IN (*.cmd) DO DISPLAY i
```

To loop setting a variable to a known set of strings:

```
FOR i IN (/usr/term/.termrc /usr/term/.termsys) DO
  IF NOT EXISTS(i)
DISPLAY "File not found: $i"
  ENDIF
ENDFOR
```

Related Topics

REPEAT, IF, SWITCH

Script Command **FTPHOST**

Purpose

To set remote host, username and password used for FTP file transfers.

Script Usage

FTPHOST *host*,*user*,*pass*

Parameters

host	network hostname for FTP transfers
user	user name on the *host*
pass	password for *user* on the *host* (May be an encrypted string.)

Description

This command is used in TERM.SYS to set the remote host, user name, and password used for FTP file transfers.

If the pass variable has been stored using the AT GET ... ENCRYPT command, or contains an ^E as the first character, TERM will automatically decrypt the password before sending it out the communications line.

Related Topics

encrypt()

Script Commands **GET - RCV - XFER - SEND**

Purpose

To send and receive files.

Script Usage

GET *filelist directory* [*options*]
GET *remotefile localfile* [*options*]

RCV *localfile* [*options*]
RCV *directory* [*options*]

XFER *filelist directory* [*options*]
XFER *localfile remotefile* [*options*]

SEND *filelist* [*options*]
SEND *localfile* [*options*]

Parameters

filelist	Is a list of files (separated by spaces) to get from or send to the remote system. Multiple files can also be specified using the wildcard characters *, ?, , or in the file name. The remote current directory is assumed unless a full path name is given. Under DOS, a drive letter and colon can be used instead of a directory. Each file specification in *filelist* can be followed by *options* valid for the command.
directory	Is the local disk drive or directory to store received files on (if GET or RCV), or the remote disk or directory where the transmitted files should be placed (if XFER or SEND). The named directory must already exist. Files will be stored in the directory with the same name as on the remote system. To receive or send files to/from the current directory, use a period (.) as the directory name. Under DOS, a drive letter and colon can be used instead of a directory. The

directory may be followed by *options* (see Options in this command description.)

localfile Is a single file name or full path name on the local system to store the received file under (if GET or RCV), or the name on the remote system to store the transmitted file (if XFER or SEND). This form of the command can only be used when transferring single files. It ignores the sending system's file name. This form must be used for MODEM7 protocol file transfers.

Description

The GET, RCV, XFER, and SEND commands allow the user to receive or transmit single or multiple files over the communications line using any of the available protocols. The commands are used as follows:

GET Is used to receive files from an unattended machine running TERM, or to request a file xfer from a "server."

RCV Is used to receive files from a system not running TERM, or when the receiving system does not know the file names to receive.

XFER Is used to transmit files to an unattended machine running TERM.

SEND Is used to transmit files to a system not running TERM.

The GET and XFER commands require that the remote TERM system to be running as a file server (-x or -1x mode), or in control mode (-c mode). This is because the GET and XFER commands request a list of files from the remote TERM, so the remote TERM must be listening for them. See the SERVER command

The syntax for the GET and XFER commands is very similar to the file copy commands available under UNIX and DOS. For instance, to get the files file1, file2 and file3 from a remote system and move them to our local current directory, type:

get file1 file2 file3 .

A list of files on the remote system is followed by the directory to put them on the local system. If the last item in the GET or XFER command is not a directory, then TERM will transfer a single file over and rename it to that file name. Only one file at a time can be transferred this way. The XMODEM protocol does not send file names across the communications line, so this second method must be used when using the XMODEM protocol.

The RCV command is used when receiving from non-TERM systems. Only the directory name or single localname is required with the RCV command. The sending system determines the files that will be sent over.

The SEND command is also used when transferring files to non-TERM systems. Notice that no directory name is used with the SEND command. The receiving system determines which directory or disk drive the transmitted files will be stored in.

With both RCV and SEND, TERM sends the local file names over the communications line using TERMCRC, WTERMCRC, ZMODEM, and KERMIT protocols. The RCV and SEND commands are not normally used with TERM to TERM communications.

When using TERMCRC, WTERMCRC, or ZMODEM protocols, TERM automatically compresses file data before it is sent out the communications line, and decompresses it after it is received. The data compression algorithm typically saves 45% of the file transfer time.

At any time, the ABORT key (typically ^C) can be pressed to abort the file transfer.

For programming support, the last file name created is kept in the system variable _filename. The variable is always set to the name of the last file on which a transfer was

attempted. This allows easier error recovery. See the examples under this command for details.

In the multiple file receive mode, file names are created on the specified directory or drive before each file is transferred. Any local file with the same name is deleted before the remote file is transferred, so be sure to specify an empty directory or drive name or have important data backed up if you do not know what remote file names are being received.

After the file name is created, data transfer begins. The following screen appears, displaying information relating to the file transfer. A description of each field on this screen can be found on the SET XFERSTAT command page.

Unless the transfer is aborted, errors only indicate line quality, as TERM will retry a packet until it gets through correctly.

After all files have been received or sent, TERM returns to terminal mode, allowing communication with the remote system. If multiple files are being received, TERM prints the file name currently being received and the total number of files transferred on termination.

Various timeout values can be changed to control how long TERM waits for initial handshaking, or how many consecutive bad packets before quitting, etc.

Options

Options are specified in parentheses following the directory or localfile. No spaces are allowed between the directory and the parenthesis. The following options may be used with the GET, RCV, XFER, and SEND commands:

(c) Convert text file. This option will cause text filecompatibility to be maintained across different operatingsystems, such as DOS to UNIX. It is useful for convertingtext files to a different operating system format whilestill providing error-checked transfer.

(f) Read *filelist* from a file. Used for XFER and SEND only. Thisoption is used to send multiple files, where each file name is read from a given file. Each file name must be includedon a separate line in that file. Thus, to send all fileslisted in the file FLIST, the option FLIST(f) is used.

(i) Ignore timeouts during ASCII and LINE transfers If alinefeed (or other line-protocol character defined by LCHAR) is never received, don't abort transfer. Instead, simply send the next line and keep going.

(q) Query. Used for XFER and SEND only. This option is only valid on multiple file transfers. It causes TERM to print

Send FILENAME (y/n):

and wait for the user to type in a response. Pressing Y will send the file. Otherwise TERM will skip the file and go on to the next. Pressing ESC will send the file and stop query mode. It sends all the following files as well. TERM sorts the list of files before querying the user, so it is easy to know how far along the sequence you are. The query option is not allowed for the GET command.

(s) Show on screen whatever data comes down the commline during transfer. This option will cause not only display of any data echoed back during full-duplex transfers, but also any prompts issued by the remote system. Used for ASCII and LINE transfers only.

(t) Translate. This option causes received files to be translated with the SET ADDCR, SET ADDLF, IGNORE INPUT, and TRANS INPUT commands before being written to disk. It is useful for receiving textfiles from a different operating system format while still maintaining error-checked file transfer.

(x) eXclude. Used for XFER and SEND only. When used with a file name, this option will cause all other files but the specified file(s) to be sent. The specified file (or multiple

files, if wildcards are used) is excluded from the file transfer. This option is useful when you want to transfer all files from a disk except a certain type (e.g., *.COM(x) will send all but the .COM files on a disk).

(z) Turn off data compression. This option disables automatic data compression during TERMCRC, WTERMCRC, and ZMODEM file transfers.

Examples

To transfer all files with the extension .TXT from directory /usr/fred on the remote system to our system's /usr/rcv directory, use

get /usr/fred/*.txt /usr/rcv

In DOS, to transfer all .TXT files from remote C: to our B:

get c:*.txt b:

To allow recovery, you can use the system variable _filename, which is set to the file currently being transferred. For instance, the following would let you log files to retry later:

```
FCREATE 1,"retry"
SETVAR files "file1 file2 /etc/termcap file5"
! get files in the list to /usr/fred directory
IFERROR GET $(files) /usr/fred
? "Get stopped at "+_filename
FWRITELN 1,"Stopped after $(_filename)"
SETVAR names midstr(files,pos(_filename,files,1),80)
FWRITELN 1,"Retry: "+names
ENDIF
FCLOSE 1
```

If for some reason file2.dat could not be sent (you didn't own it or it wasn't there), _filename would be file2.dat and the file retry would contain the following two lines:

```
Stopped after file2.dat
Retry: file2.dat /etc/termcap file5.dat
```

Examples of directory:

. Receive files on current directory.

/usr/greg Receive files on directory /usr/greg.

/usr/greg(c) Receive files in directory /usr/greg and do text file conversion

a: Receive files on drive A: (DOS).

The following are other examples of filelist:

letter.txtSend the file letter.txt only

*.txt Send all files with the extension .TXT

b:*.* Send all files on drive B: (DOS)

*.doc(q) Send all .DOC files approved by user

ap*.*(x) Send all files not starting with AP

Related Topics

PROTOCOL, XPROT, RETRIES

Script Command **HANGUP**

Purpose

To hangup phone and logout from remote system.

Script Usage

HANGUP

Description

This command causes TERM to execute the LOGOUT command (if defined) and hangup the phone, disconnecting any datalink previously established.

If the connection is a direct async connection without a modem, the HANGUP command will only execute the LOGOUT command, then return to terminal mode.

Related Topics

CONNECT, ANSWER, DIALER, MODEM, LOGOUT, SESSION

Script Command **HELP**

Purpose

To provide online help.

Script Usage

HELP ["*topic*"]

Parameters

topic a string that is a keyword to lookup in the help system.

Description

Brings up the Help System.

The HELP command now uses the Windows help system to provide on-line help in Windows.

For help operation, see your Windows documentation or the "How to Use Help" option in help.

The *topic* parameter provides similar function to the help system's search option.

Script Commands **IF - IFERROR - ELSE - ELSEIF - ENDIF**

Purpose

To provide conditional execution in script files.

Script Usage

```
IF condition
  statements
ELSE
  statements
ENDIF
```

or

```
IF condition
  statements
ELSEIF condition2
  statements
ELSEIF condition3
  statements
ELSE
  statements
ENDIF
```

or

IFERROR *command*
  statements
ELSE
  statements
ENDIF

Parameters

condition Is an expression that evaluates to a TRUE/FALSE value. See expressions.
command Is any valid TERM script command. If the command returns an error status, an abort will not be triggered. Instead, IFERROR will be true.

Description

This command allows conditional statement execution in script files. The statements between the IF and ENDIF are evaluated only if the condition is true. Optional ELSE and ELSEIF clauses allow for execution of statements if the condition was false. IF statements can be nested to any level, with the following rules:

* Each IF must have a corresponding ENDIF.
* IF/ENDIF pairs must both be within the same script file.

The IFERROR statement is used to catch errors while executing any script command. The command, such as XFER or CONNECT, is run and the error status returned. If an error occurred, the IFERROR acts like a statement with a true condition.

NOTE IFERROR XFER does not detect an error if the file does not exist. To avoid this condition, use the function exists() to check for existence before transferring the file.

Examples

To catch errors while trying to connect to another system named UNIXA:

```
IFERROR CONNECT unixa
  DISPLAY "Can't connect to UNIXA system"
  RETURN
ENDIF
```

Related Topics

REPEAT, FOR, SWITCH, RETURN

Script Command **IGNORE**

Purpose

To ignore characters on received or transmitted data.

Script Usage

IGNORE [INPUT or OUTPUT] *ignore string*...
IGNORE RESET

Parameters

ignore string Specifies the range of characters to ignore on data received or transmitted on the communications line. The string must be enclosed in quotes if spaces are used, and the dash (-) character may be used to denote a range of characters (i.e., a-f denotes the string abcdef). See Specifying Strings.

Description

This command is used to ignore a specified set of characters received or transmitted on the communications line. Files moved by the SEND or RCV commands are modified only if the (t) option is given with the SEND or RCV command.

The user selects whether to ignore incoming data, outgoing data, or both.

NOTE TERM first checks for IGNORE characters, then TRANSlate characters, then ADDCR/ADDLF. Thus, characters specified by the IGNORE command and also specified for translation will be ignored, not translated. However, characters TRANSlated to an IGNORE character will not be ignored.

IGNORE RESET resets any previously executed IGNORE commands.

Examples

To ignore all numeric characters received, use

ignore input 0-9

To prevent TERM from sending out any control characters, use

ignore output -

Related Topics

SET ADDCR, SET ADDLF, trans, File Transfer

Script Command **INTER**

Purpose

To put TERM in interactive script mode.

Script Usage

INTER

Description

This command puts TERM in interactive script mode. TERM's interactive script prompt is a colon (:). Any script command entered will be executed when you press return on a line. Type END and press ENTER to stop interactive script mode. This command provides a useful debugging tool and script writing aid.

Script Command **LCHAR**

Purpose

To define ASCII line send protocol character.

Script Usage

LCHAR *charnum*

Parameters

charnum Is the ASCII value of the new line protocol character to use with LINE transfers, other than the default linefeed character decimal 10.

Description

During LINE sends, TERM waits for the receiving computer to respond with the line protocol character (by default a linefeed character, decimal 10) to request sending of the next line.

The line protocol character can be changed with the LCHAR command. Lines are defined by up to 127 characters followed by a carriage return and/or line feed character.

Related Topics

CDELAY, File Transfer, XPROT, ITIME

Script Command **LEARN**

Purpose

To start and stop learn mode.

Script Usage

LEARN START *filename*
LEARN STOP

Parameters

filename Is the name of the file to be created from the learn mode session. If this file exists, it will be overwritten.

Description

This command is used to START and STOP the learn mode feature within a script.

Learn mode is only active during terminal emulation sessions. This means that it will not capture TERM menu selections or TERM commands.

The learn mode parser attempts to capture the most significant portions of what is sent from the remote system, which is usually the last 6 or so characters before an answer from the local computer.

Examples

```
.
LEARN START foo.cmd
  terminal "done"
LEARN STOP
.
```

The above example will create the file foo.cmd and record everything sent and parse everything received during terminal mode. The terminal command will continue in terminal mode until it receives the string "done".

Related Topics

TERMINAL

Script Command **LOGFILE**

Purpose

To specify a log file name for error logging.

Script Usage

LOGFILE *filename*[OVERWRITE]
LOGFILE CLOSE

Parameters

filename Is the name of the logfile to open. The file name is not quoted.

Description

The LOGFILE command specifies a logfile, and by default appends data to it using the LOG command. If the OVERWRITE option is specified, TERM will overwrite the file if it already exists.

This command is useful in automated scripts where it is desired to keep a disk file log of activity. The LOG command writes the string to a file in operating system independent format.

The logfile can be closed by specifying a new logfile, or by executing the LOGFILE CLOSE command. TERM will also close a logfile on exit.

Examples

To overwrite the logfile ERRLOG:

LOGFILE errlog OVERWRITE

Related Topics

LOG, FOPEN, SET LOG

Script Command **LOG**

Purpose

To write a string into the last opened log file.

Script Usage

LOG *string*

Parameters

string Is the character string expression to log to the current logfile.

Description

This command is useful in automated scripts where it is desired to keep a disk file log of activity. The LOG command writes the string to a file in operating system independent format.

Examples

To overwrite the logfile ERRLOG:

LOGFILE errlog OVERWRITE
LOG "Error #3: Can't happen"

Related Topics

LOGFILE, FOPEN, SET LOG

Script Command **DIALOG**

Purpose

To execute a dialog string.

Script Usage

DIALOG *string*

Parameters

string a string containing the dialog string commands and other text.

Description

The DIALOG strings allow sending data and waiting for data to and from the communications line. Each character of the DIALOG string is sent unless a special control character is encountered, just as in LOGIN strings. DIALOG uses the same control characters as the LOGIN command.

Related Topics

RESTART, LOGIN, LOGOUT, STARTSERVER, STOPSERVER, Dialog Strings

Script Command **LOGIN**

Purpose

To specify an automatic login sequence.

Script Usage

LOGIN *string*

Parameters

string Is the coded automatic login sequence.

Description

The LOGIN command specifies a login sequence with a remote computer. TERM automatically executes the login sequence after receiving the "CONNECT" from the modem, or after three seconds for direct serial connections.

If desired, LOGIN or LOGOUT may be specified without the string parameter for immediate execution.

Examples

If your user id was guest and your password was foo, and you wanted to login to a system that sent a login: prompt followed by a Password: prompt, you could use the following LOGIN string to autologin:

LOGIN "^L06^T01^Sgin:^S^Wguest^Sword:^S^Wfoo"

This means try up to six times; send a return, check for login:. If after one second, it is not seen, try again. When seen, wait one second, send guest followed by return. Then wait for word:. If not seen, start all over. Otherwise wait one more second, then send foo followed by return.

Related Topics

CONNECT, RESTART, LOGOUT, STARTSERVER, STOPSERVER, Dialog Strings

Script Command **LOGOUT**

Purpose

To specify an automatic logout sequence.

Script Usage

LOGOUT *string*

Parameters

string Is the coded automatic logout sequence.

Description

The LOGOUT command specifies a sequence with a remote computer. TERM automatically executes the LOGOUT sequence before the HANGUP command disconnects the datalink. If desired, LOGOUT may be specified without the string parameter for immediate execution.

Examples

LOGOUT "exit"

The above will transmit the word exit to the host followed by a carriage return. This is usually sufficient for a UNIX connection.

See Also

dialog strings

Related Topics

CONNECT, HANGUP, RESTART, LOGIN, STARTSERVER, STOPSERVER, Dialog Strings

Script Command **MEMLIST**

Purpose

To display all memory variable names and values.

Script Usage

MEMLIST [L]
MEMLIST SYS

Description

This command lists the currently defined memory variables, as in the following sample:

Name	**Type**	**Value**
X	numeric	1
Y	string	"foobar"

If the L (long) option is specified, TERM will print out the contents of indexed variables as well. The list can be stopped at any time by typing CTRL+S (^S), and terminated with a CTRL+C (^C).

The SYS option displays all of the named TERM internal variables.

Related Topics

SETVAR, READ, CREAD, DIMSTR, sysvar(), Variables

Script Command **MENU ENTRY**

Purpose

To create items for the pulldown list portion of the Main Menu.

Script Usage

MENU *id* ENTRY *title* EXECUTE *command*

Parameters

id The id of the "parent" menu item.

title title of the option shown on the menu. A dash (-) creates a separator bar in the pulldown list.

command a script statement in quotes, may be any valid script language command including DO file@proc

Description

Creates entries for TERM's pulldown menu lists.

Internally the items of the list are given the value of the parent menu item plus its location in the list. Item 1 of menu 100 would have an implicit ID of 101, item 5 would be identified as 105.

NOTE The ampersand character (&) in the *title* will cause the next letter to be underlined, and to be the short cut key for selecting the option. To display an ampersand character two ampersands are required (&&).

Examples

The statements to create the file entry and pull down in the standard user interface follow. See WTERM<language>.CMD for the full menu declaration.

```
MENU 100 OPEN "&File"
MENU 100 ENTRY "&Open Configuration..."   EXEC "do wtermus@w_file_load"
MENU 100 ENTRY "&Save Configuration"      EXEC "do wtermus@w_file_save"
MENU 100 ENTRY "Save Configuration &As"   EXEC "do wtermus@w_file_saveas"
MENU 100 ENTRY "-"
MENU 100 ENTRY "&Print Screen"            EXEC "screen print"
MENU 100 ENTRY "&Edit Text File..."       EXEC "do wtermus@w_file_edit"
MENU 100 ENTRY "Change &Directory..."     EXEC "do wtermus@w_file_chdir"
MENU 100 ENTRY "Erase &File..."           EXEC "do wtermus@w_file_erase"
MENU 100 ENTRY "-"
MENU 100 ENTRY "E&xit"                    EXEC "do wtermus_w_file_exit"
```

Related Topics

MENU OPEN

Script Command **MENU OPEN**

Purpose

To create a menu item on TERM's Main Menu.

Script Usage

MENU *id* OPEN *title*

Parameters

id numeric menu id. In the standard user interface menu item ids begin at 100 and increment by 100. 900 is the help menu.

title title of the option shown on the menu

Description

Creates entries for TERM's main menu. MENU ENTRY creates the items in the pull down list.

NOTE The ampersand character (&) in the *title* will cause the next letter to be underlined, and to be the short cut key for selecting the option. To display an ampersand character two ampersands are required (&&).

Related Topics

MENU ENTRY

Script Command **MKDIR**

Purpose

To create a new disk directory.

Script Usage

MKDIR *newdir*

Parameters

newdir Is the new directory name. *newdir* is not a quoted string.

Description

This command is used to make a new directory from within TERM's script language. Making a directory which already exists will cause an error.

Examples

To make the directory maketerm under the current directory:

```
MKDIR maketerm
```

To make the directory /usr/term/final under the root directory (/usr/term must already exist):

```
MKDIR /usr/term/final
```

To make a directory only if there is not already a file or directory by that name:

```
SETVAR foo "/usr/term/final"
IF !EXISTS(foo) & !ISDIR(foo)
MKDIR $(foo)
ELSE
  ? "There is already a file/dir named that!"
ENDIF
```

Related Topics

TYPE, RENAME, ERASE, COPY, RMDIR, RUN

Script Command **MODE**

Purpose

To select terminal emulation communications modes.

Script Usage

MODE [FULL, HALF, DUMP, MNEMONIC, or CONTROL]

Description

This command allows the user to specify one of six possible communications modes during terminal emulation. Different communications modes can be selected depending upon the type of remote system with which the user is communicating. The following is a description of each:

FULL

Full duplex. Full duplex is normally used when communicating with multi-user computer systems running UNIX or VMS, as well as large mainframes or timesharing systems like The Source, or even local hobbyist bulletin boards. This mode causes TERM to act like a dumb terminal and send any data typed over the communications line, and display any data received. Full duplex emulation defaults when TERM is first started.

HALF

Half duplex. Half duplex operation is just like full duplex, except that characters typed are immediately displayed on the screen before being sent over the communications line. This mode is seldom used except for communicating with older time-sharing systems. Half duplex can also be selected by the command line option -h.

If in doubt as to whether to select Half or Full duplex when connected to a remote computer system, first try full duplex; if the characters aren't being displayed on the screen after typing, then switch to half duplex.

DUMP

Dump mode. Dump mode will display two-digit hexadecimal characters for any data received over the communications line during terminal emulation. Data capture with the CAPTURE command is not operational in this mode. Dump mode is very useful for verifying exactly what data is being received from a communications line.

MNEMONIC

Mnemonic dump mode. This mode displays any non-printable incoming data as three letter codes enclosed in brackets (i.e., CTRL A (hex 01) would be displayed as <SOH>). If the high bit is set for any incoming character, a plus sign (+) is displayed prior to its 7 bit value. This mode is very useful for using TERM as a comm line analyzer.

CONTROL

Control character dump mode. This mode displays any non-printable incoming data as a caret followed by the character (i.e. CTRL A (hex 01) would be displayed as ^A). If the high bit is set for any incoming character, a plus sign (+) is displayed prior to its 7 bit value. This mode is also very useful for using TERM as a communications line analyzer.

Examples

The following example shows what you will see in the various dump modes, given a specified serial bit pattern:

Serial Data You See

00000001--Dump mode---------- 01
00000001--Mnemonic Dump------- H
00000001--Control Char Dump-- ^A

01000001--Dump mode---------- 41
01000001--Mnemonic Dump------ A
01000001--Control Char Dump-- A

11000001--Dump mode---------- c1
11000001--Mnemonic Dump------ +A
11000001--Control Char Dump-- +A

Related Topics

EMULATE, PROTOCOL, TERMINAL

Script Command **MODEM**

Purpose

To set default modem type and dialer script.

Script Usage

MODEM *scriptname*

Parameters

scriptname Is the name of a pre-built modem script, usually residing in TERMDIR. If *scriptname* is set to HAYES, TERM's internal dialer is used.

Description

This command is used to interface with any modem that is not Hayes-compatible. TERM may send specific modem sequences during the CONNECT, HANGUP, and ANSWER commands. The MODEM command specifies the name of a script file to call for each of these commands.

The script file must contain three procedures, named call, hangup and answer. TERM then calls the script *scriptname* with the following parameters each time for the CONNECT, HANGUP and ANSWER commands: CALL

DO *scriptname*@call number redials dtimeout

where number is the number from the NODE statement, redials is the current REDIALS setting, and dtimeout is the current DTIMEOUT setting.

HANGUP
DO *scriptname*@hangup ddtr

where ddtr is the current SET DDTR setting.

ANSWER
DO *scriptname*@answer type

where type is one of the strings OFF, WAIT, AUTO, or NOW.

Immediately after the MODEM command is specified, TERM executes the *scriptname* with the following:

DO *scriptname*

That is, the outer loop code is executed.

The TERM distribution comes with several pre-built modem scripts for popular non-Hayes modems. These scripts are all files named MMD*.CMD. If you feel like building your own, we suggest you look at these examples first.

To reset TERM to use its internal HAYES dialer, use:

MODEM hayes

Examples

To set TERM to call the script MDMNEW.CMD:

MODEM mdmnew

Related Topics

CONNECT, NODE, DIALER, ANSWER, Answering, Connecting, HANGUP, mdmstat()

Script Commands **NETPORT - NETVTNAME - NETDIALOG**

Purpose

To configure MSNET/NetBIOS port number, virtual terminal name and handshake.

Script Usage

NETPORT *n*
NETVTNAME *string*
NETDIALOG *string*

Parameters

n	Is the numeric digit equal to the ASCII value of the requested port number.
string	Is the coded virtual terminal name or handshaking string.

Description

DOS/Windows version only. These commands allow maximum flexibility in configuring MSNET/NetBIOS virtual circuits. TERM's defaults should work with the majority of systems. NETPORT sets the 16th character of the MSNET/NetBIOS name. It is specified as a numeric digit equal to the ASCII value of the requested port number. It also specifies the telnet or rlogin port number on non-standard TCP/IP stacks.

NETVTNAME controls the creation of the virtual terminal server name from the name passed to TERM on the command line. It uses the standard "C-style" printf strings (i.e., %s matches the user specified name). For example, "%s.VT" means take the user specified

nodename and follow it with a '.VT' to create the virtual terminal server name for the MSNET/NetBIOS call command.

NETDIALOG controls the handshaking required after the MSNET/NetBIOS call command completes in order for the VT server to grant a virtual terminal line. Its format is similar the LOGIN command with an additional control character. ^Z sends a null character. As an example, the string "^SiVTS^S^FiPC1^Z" means:

^SiVTS^S Search for the string "iVTS".

^F Flush further input.

iPC1^Z Send "iPC1" followed by a null character.

Related Topics

NETSTIME, Dialog Strings

Script Command **NOERROR**

Purpose

To inhibit script termination for one command.

Script Usage

NOERROR *command*

Parameters

command Is any valid TERM script command.

Description

This command is used to execute a script command from an automated script which doesn't want to abort due to an error condition.

The IFERROR command is used to branch on the result of a script error. Normally the NOERROR command is used with commands like RENAME or ERASE (i.e. in cases where it doesn't matter if the command failed). An error message will still be displayed, even though the error itself is no longer fatal.

Examples

To execute the RENAME command regardless of its results:

NOERROR RENAME foo fee

Related Topics

EXECUTE, IFERROR, eval()

Script Command **EXECUTE**

Purpose

To inhibit script termination for one command.

Script Usage

EXECUTE *str*

Parameters

str a string variable containing any valid TERM script command.

Description

This command is used to execute the script command contained in a variable from an automated script which doesn't want to abort due to an error condition.

The variable for the EXECUTE command cannot be macro-expanded; therefore the contents of the string may not contain an expression that requires macro-expansion.

The IFERROR command is used to branch on the result of a script error. Normally the EXECUTE command is used with commands like RENAME or ERASE (i.e. in cases where it doesn't matter if the command failed). An error message will still be displayed, even though the error itself is no longer fatal.

Examples

To execute the RENAME command regardless of its results:

SETVAR *str* "RENAME foo fee"
EXECUTE *str*

Related Topics

NOERROR, IFERROR, eval(), Strings

Script Command **NETSTIME**

Purpose

To set the NetBIOS/MSNET send timeout value.

Script Usage

NETSTIME *n*

Parameters

n number of 1/2 second intervals to wait for a network response before timing out.

2 is the default value (wait 1 second for a responsebefore timing out).

0 is the value for never time out.

Description

This command is used in TERM.SYS to set the NetBIOS/MSNET send timeout value. If the connection drops during high network traffic, you should increase this value.

Script Command **NETWORK**

Purpose

To set the network type.

Script Usage

NETWORK *str*

Parameters

str string containing the name of a supported network.

Available networks:

bapi	3Com TCP BAPI
bw	Beame & Whiteside BW-TCP
com1	com1
com2	com2
com3	com3
com4	com4
com5	com5
com6	com6
int14	Interrupt 14

msnet	Microsoft Networks
netbios	NetBIOS
nul	nul
opennet	Intel OpenNET
opennetb	Intel OpenNET b
pci	Locus PC-Interface
pcnfs	Sun Microsystems PC-NFS
pctcp	FTP Software PC/TCP for DOS
telapi	Novell Telapi
ub	Ungermann Bass Net One
xenixnet	SCO XENIX-NET

Description

This command, used in TERM.SYS, sets the network type to be used with TERM. TERM may load the async driver and 1 network driver. The network driver may support up to 9 active network sessions.

Purpose

To define event-catching procedures.

Script Usage

```
ON KEY key
ON ERROR
ON ABORT
ON CARRIER  (DOS ONLY)
ON NOCARRIER  (DOS ONLY)
  .
 (statements to be executed when ON condition is satisfied)
  .
ENDON
```

Parameters

key	Is an expression that evaluates to any ASCII character, from 0 to 127 decimal (0x01 to 0x7f hex.)

Description

These command pairs are used to specify actions to be taken when TERM detects that a specified event has occurred, such as a certain key is pressed, a time delay has expired, or carrier is lost. These commands are valid only when executed from within a script file. This means that you CANNOT set up an ON command, enter terminal mode, and still expect it to be in effect. The ON commands must also be the first lines in the script file.

Groups of valid TERM commands are placed between the ON and ENDON statements; these commands are not executed until the given ON condition takes place.

TERM will continue execution of the script file at the point it was interrupted. ON statements only work in the script file in which they are defined. Executing a DO statement to another batch file, or RETURNing from a script to command mode will terminate any ON statements.

The following are descriptions of each ON statement:

ON KEY *key*
This command defines a group of statements to be executed whenever a specified key is pressed. Any number of ON KEY statements may be in effect for different keys at the same time. When the specified character is pressed, the current command is suspended and the ON KEY procedure starts executing.

When using the TERMINAL script command during terminal emulation, the ON KEY character is not sent to the remote system. If an ON KEY procedure is defined for the key that is your command character, ON KEY will take precedence, allowing a terminal emulation without giving direct control to the user. The ON KEY statement is useful for defining single-character logins, logouts and other functions.

The ON KEY statement will only work within a first-level script file (i.e. executed from a command-mode DO statement or executed as an argument to TERM). It will not work in nested scripts or PROCs.

ON CARRIER
This command defines a set of commands to be executed if carrier changes from OFF to ON status on pin 8 of your system. It currently works under DOS systems only.

ON NOCARRIER
This command defines a set of commands to be executed if carrier changes from ON to OFF status on pin 8 of your system. It currently works under DOS systems only.

ON ERROR
This command defines a set of commands to be executed if a script command returns an ERROR status.

bON ABORT
This command defines a set of commands to be executed if a script command be aborted by the ABORT character (usually ^C). See the SETKEY command page for information on changing the ABORT key.

Examples

The following script section would catch abort attempts:

```
ON ABORT
  DISPLAY "Logging out now..."
  XMIT "logout"
  HANGUP
ENDON
```

Related Topics

RETURN, QUIT, TERMINAL,

Script Command **PAGES**

Purpose

To set the number of available screen memory pages in an emulation.

Script Usage

PAGES *n*

Parameters

n	Number of screen pages to allocate (range from 1 to 6).

Description

This command is used to define the number of pages of screen memory to allocate on session startup.

The following emulations support page memory:

AT386, SCO ANSI, VT100/VT52, VT220, VT320, Wyse 50, Wyse 60,

Related Topics

EMULATE

Script Command **PASSWORD**

Purpose

To specify the password for use with the ^P character in DIALOG and LOGIN strings.

Script Usage

PASSWORD *str*

Parameters

str	password for use in logging onto a system.

Description

This command is used in .TAP files to specify the password to use for the ^P character in DIALOG or LOGIN strings. If the *str* variable starts with an ^E, or was created using the AT GET ... ENCRYPT command, it will be automatically decrypted before being sent out the communications line.

Related Topics

Dialog Strings, LOGIN, encrypt()

Script Command **PAUSE**

Purpose

To pause script file execution.

Script Usage

PAUSE [*message*]

Parameters

message Is an optional message to be displayed on the console. Message is not a quoted string and cannot include leading spaces.

Description

This command is used to temporarily pause the execution of script files. It waits for a key to be pressed by the operator before continuing. Typing a CTRL+C (^C) will abort the command file in process.

Upon default execution of this command, TERM displays the message "Strike a key when ready..." and waits for a key to be depressed before continuing. If message is specified, that message will be displayed instead.

Examples

Since messages cannot be quoted, the only way to indent a message is to output preceding spaces with the DISPLAY (?) command and use a semicolon so that both lines are printed together:

```
? " ";
PAUSE This is an indented message.
```

Related Topics

AT, CREAD, READ, DISPLAY

Script Command **PORT**

Purpose

To specify default communications line or network access method.

Script Usage

PORT *portlist*

Parameters

portlist The device to connect to at startup. A *portlist* cannot contain ANY spaces.

Description

This command is used in configuration files to specify TERM's communications access method.

On DOS, Port specifies the method of access. COM1: through COM2: are async connections and need no other settings. For network connections the network command must be used to define the network type in use. Port will then set the access type: telnet, rlogin, telnete, telnetn, telnetb, telneti and so forth.

Examples

To set an async connection:

PORT "com2:"

To set a network connection:

PORT "telnet"

Related Topics

NETWORK, DEVPREFIX, FLAGS, EDITOR, MODEM

Script Command **DEVPREFIX**

Purpose

To specify the OpenNET device prefix.

Script Usage

DEVPREFIX *str*

Parameters

str	Is a quoted string containing the prefix TERM should prepend to items in the portlist.

Description

This command, used in the configuration files, set OpenNET node device prefix which is prepended to the portlist.

Examples

```
DEVPREFIX  "//century"
```

Related Topics

PORT, FLAGS, EDITOR, MODEM

Script Command **PRINTER**

Purpose

To setup output options for virtual printer requests.

Script Usage

PRINTER NONE
PRINTER DEVICE *dev* [*options*]
PRINTER DISK *filename* [*options*]
PRINTER SPOOL [*spooler*]
PRINTER CAPTURE
PRINTER CLOSE
PRINTER FLUSH
PRINTER ON
PRINTER OFF

Parameters

dev	Specifies the device to to which to route printer requests.
filename	Specifies the path name of the file to use when routing printer requests to a disk file.
spooler	Is an optional specification for the name of the line printer spooler to use when routing printer requests to the spooler. If omitted, the standard line printer spooler is used.

Description

This command specifies what action TERM should take when any requests for printing are encountered.

After a print request is seen, the status of the virtual printer is ON. It remains on until a PRINTER CLOSE or PRINTER OFF command is executed, a transparent print off sequence is sent by the host, or TERM is exited. TERM does not run the printer spooler or open a disk printer file until the first print request is seen.

The following are detailed descriptions of each of the PRINTER options:

NONE
Ignore all print requests.

DEVICE
Similar to Disk, except overwrite and append have nomeaning.

DISK
Open a disk file when a print request is seen. The disk file will remain open until a PRINTER CLOSE or another PRINTER command is given. If the option APPEND is given, TERM will append to the given file. If the OVERWRITE option is given, TERM will overwrite the filewithout asking. By default, TERM will ask the user whether an existing file should be overwritten or appended to.

If the *filename* contains a number sign (#) character, TERM will generate a unique filename by substituting digits 00 through 99 until a new file name is found.

SPOOL
Start the system line printer spooler when a print request is seen. The spooler will remain executing until a PRINTER CLOSE or anotherPRINTER command is given. Another printer spooler may be used by specifying the spooler argument.

CAPTURE
Route print requests to the data capture system. Print requests will be saved no matter how the data capture has been specified. This is useful to save both data capture and printer requests to the same file. See the CAPTURE command page for details on capture requests.

CLOSE
Close any open disk capture file or line printer spooler. They are also closed automatically by TERM before quitting.

ON
Restarts printer capture operation. If capture has been turned OFF, PRINTER ON will simply re-open the same disk file. However, ifcapture has been CLOSEd, PRINTER ON will increment the file name and start a new file (assuming a disk file containing a number-sign character in the name was previously being used for printer capture).

OFF
Stops printer capture operation. Unlike CLOSE, OFF does not actually close the file, but simply temporarily stops adding data to it. When a subsequent PRINTER ON command is received, data will be appended to the originally OPENed file.

NOTE If the CLOSE command is not used, but the file is viewed by the TERM TYPE command, an editor, or the DOS TYPE command, the file will seem to be incomplete. The PRINTER CLOSE command MUST be used before doing anything with the file, including COPYing it.

Options

The file name can be followed by options as follows:

APPEND Force a file to append to the named file.

OVERWRITE Force a file overwrite if the named file already exists.

FLUSH This option will help prevent file corruption becauseof open filesduring a system crash. This causes disk files to be written, closed, and re-opened after a PRINTER OFF is executed. When the printer device is a non-disk file, such as printing direct to LPT1: in DOS or to /dev/lp on UNIX, this option will ensure that all buffered data is flushed rather than kept in TERM's buffers.

Examples

To send all printer activity to be appended to the file printer.sav:

PRINTER DISK printer.sav APPEND

To send all print activity to unique disk files named in sequence from pr00.fil to pr99.fil:

PRINTER DISK pr#.fil

To send printer output to the LPT1: parallel print port on a DOS system:

PRINTER DEVICE lpt1:

At higher baud rates the printer may not be able to keep up with the communications line. In that case, XON/XOFF flow control is used down the commline (if protocol XON is set) to tell the system sending the information to stop until the buffer is no longer full. If protocol XON is not set or if the remote system doesn't respond to XON/XOFF flow control, the buffer will be overrun and you will lose information. To correct this, print to a disk file and use the DOS PRINT command after leaving TERM to print the file to the printer. Also the print buffer may be configured on the command line with the -p command line option

Related Topics

CAPTURE

Script Commands **PROC - ENDPROC**

Purpose

To define a procedure within a script file.

Script Usage

PROC *procname*
.
statements to be executed when this procedure is called
.
ENDPROC

Parameters

procname Is the procedure name. Up to 10 alphanumeric characters are allowed.

Description

The PROC statement defines a group of statements to be executed as a group within a script file. When the ENDPROC statement is reached, control is returned to the calling procedure. The commands between the PROC and ENDPROC statements are never executed until a DO command calling that procedure is executed.

Procedures are called using the @ character in a DO statement. For example:

```
DO foo ! call script "foo", outer code
DO foo@proc1 ! call script "foo", proc "proc1"
DO @proc1
! call proc "proc1" in current script
```

Any number of procedures may be included in a script file. When a script file is called without specifying a procedure name (as in DO foo above), only those statements outside of any PROC-ENDPROC grouping will be executed. When a script file is called by specifying a procedure name (as in the other two examples above), only those statements within the named PROC will be executed. Control is then returned to the statements following the DO.

NOTE All PROC statements MUST have an ENDPROC. Their procedures cannot be nested.

Examples

To call the local procedure PROCLOC with parameters "A", "B", and "C":

DO @procloc A B C

Related Topics

DO, RETURN, MODEM

Script Command **PROTOCOL**

Purpose

To set communications line protocol used during ASCII file transfers.

Script Usage

PROTOCOL [XON, ETX, NONE]
PROTOCOL [QUIET ON|OFF, DCD ON|OFF]

Description

The following three options are used to control outward data flow. They are mutually exclusive.

Xon
Use Xon/Xoff protocol to control outward data flow. This is the default protocol when TERM is first executed. Xon/Xoff also selects kernel Xon/Xoff support for receiving files at high baud rates using the data capture commands and the transparent printer commands.

Etx
Use Etx/Ack protocol to control outward data flow. This is useful for sending data to older printers which require this type of handshaking. If you need to RECEIVE data from a computer that expects ETX/ACK handshaking, you can do the following:

```
WRUCHAR "^C"
!Set who are you char to ETX
WRU "^F"
!Set answerback to ACK
```

When the computer sends an ETX, TERM will reply with an ACK, and the computer will think a printer is there and continue printing.

NONE
No protocol. Comm line data is not interpreted in any way. (However, you can slow down a transfer, even when NONE is selected, by using the LDELAY and/or CDELAY commands.)

The following two options are toggle options. The first time the option is selected, it is turned ON. Selecting the option a second time in the same session toggles it OFF.

DCD
(DOS only.) This option is toggled on or off each time it is selected. When on, TERM checks if the Carrier Detect signal (typically pin 8) from the modem ever goes from on to off (indicating a phone connection just terminated). If it detects such a switch, it closes and writes the capture file if a CAPTURE command was in effect.

QUIET
This option is toggled on or off each time it is selected. When on, TERM suppresses display of captured data during a file capture command. This is useful for capturing binary data or data at high baud rates.

Examples

To set the protocol to Etx/Ack, use

PROTOCOL ETX

Related Topics

XPROT, File Transfer, CAPTURE

Script Command **QUIT**

Purpose

To quit TERM and return to the operating system.

Script Usage

QUIT *exitval*

Parameters

exitval Is the value to return to the calling program. This value must be 0 to 255.

Description

This command is used to quit TERM. The system will be returned to the operating system.

Any open capture file is automatically written to disk and closed before TERM exits to the operating system.

Related Topics

RETURN

Script Commands **READ - READN - READINT**

Purpose

To read characters from the keyboard to a variable.

Script Usage

READ [*string*,] *variable*
READN [*string*,] *variable*
READINT [*string*,] *variable*

Parameters

string	Is an optional prompt string.
variable	Is the variable to create and load with the keystroke.

Description

This command is used to read data from the keyboard and store it in a memory variable. The READ command creates a string-type variable; READINT creates a numeric-type variable. If the optional string is given, that string is displayed as a prompt before the read is initiated. READN is the same as READ, except characters typed are not echoed to the screen. This is useful for entering a password into a script, without displaying the password.

The READ commands can be aborted by typing the ABORT character (usually ^C). If this is not desired, the SET ABORT OFF command must be entered before the READ command in order to inhibit script aborts by user-typed ABORT characters.

Examples

To prompt for a number to be entered:

READINT "Please enter desired number:",n

To prompt for a login id:

READ "Enter login account number:",login

To disallow aborts during a read:

```
SET ABORT OFF
READ X
SET ABORT ON
```

Related Topics

DISPLAY, MEMLIST, SETVAR, CREAD, PAUSE, strconv()

Script Command **Redraw**

Purpose

To redraw the emulation screen.

Script Usage

REDRAW

Description

Redraws TERM's terminal emulation screen. This command is typically used after the RUN command to restore the terminal screen contents.

Related Topics

RUN

Script Command **RELEASE**

Purpose

To deallocate and release unused variables

Script Usage

RELEASE *variable* [,*variable*]
RELEASE *

Parameters

variable Is the name of a variable to be released.

Description

This command releases a list of variables that are no longer required. All variables can be released by using an asterisk (*) as the parameter. This command cannot have a comment (!) included on the same line.

Once variables are released, their memory may be used by other procedures within TERM.

Examples

To release all variables in use:

```
RELEASE *
```

To release the variables count, lister, and x:

```
RELEASE count,lister,x
```

Related Topics

DIMSTR, SETVAR, MEMLIST, COMMENT, VARIABLES

Script Command **REMOTE**

Purpose

To execute a TERM command on a remote system.

Script Usage

REMOTE *command*

Parameters

command Is the entire command line to be executed on the remote system.

Description

This command allows the user to send a command to a remote system running TERM. Any valid TERM script command may be given. The remote TERM must be running in server mode or be in control mode.

Examples

To tell a remote TERM to quit:

```
REMOTE quit
```

To ask if the remote file foo exists, and if it does, get it from the default directory:

```
SETVAR file "foo"
REMOTE XMIT "(exists("$(file)"))"
CREAD ft, 10
IF _timeout | (ft!="true" & ft!="false")
 ? "Check failed... make sure remote system"
 ? "is running 'wterm -x'"
ELSE
```

```
IF $(ft)

  GET $(file) .
ELSE
  ? "Remote file does not exist"
ENDIF
ENDIF
```

Related Topics

RUN

Script Command **RENAME**

Purpose

To rename a file.

Script Usage

RENAME *oldname newname* [OVERWRITE]

Parameters

oldname	Is the original file name.
newname	Is the new file name.

Description

This command is used to rename a file from within TERM's script language. If the file *oldname* does not exist, TERM reports an error. If you do not have privileges for the file, the directory the file is in, the destination file name (in the case of the OVERWRITE option), or the destination directory (not valid for DOS), TERM reports an error.

If the OVERWRITE option is specified, the *newname* file will be overwritten if it already exists. Otherwise, TERM reports an error if the *newname* file already exists.

The *oldname* and *newname* file names are not quoted.

Under DOS, the *oldname* and *newname* files must reside under the same directory.

Examples

To rename the file one to the file two:

```
RENAME one two
```

Related Topics

COPY, ERASE, MKDIR, RMDIR, CD, TYPE, RUN

Script Command **REPEAT**

Purpose

To provide looping control in script files.

Script Usage

```
REPEAT
 .
 (these commands execute until expr is true)
 .
UNTIL expr
```

Parameters

expr	Is an expression that evaluates to a TRUE/ FALSE value.

Description

This command pair allows controlled looping within script files. The statements within the REPEAT/UNTIL pair will always execute once. Statement looping stops as soon as the *expr* is evaluated as TRUE. REPEAT loops may be nested up to 20 deep, subject to the following rules:

* Each REPEAT must have a corresponding UNTIL.

* Each REPEAT/UNTIL pair must be in the same script file or procedure.

See Rules for Expressions

Examples

To count from 1 to 10 on the screen:

```
SETVAR j 1
REPEAT
 DISPLAY j
 SETVAR j j+1
UNTIL J  10
```

Related Topics

IF, SWITCH, FOR, PROC, DO, RETURN

Script Command **RESTART**

Purpose

Restartable file transfer control.

Script Usage

RESTART *n*

Parameters

n Is a number representing the number of retries

Description

The RESTART command controls the number of automatic file restarts TERM will attempt before aborting the execution of the XFER or GET commands. Setting RESTART to 0 turns off automatic file restarts, which is the default. The TERMCRC, WTERMCRC or ZMODEM protocols are the only protocols that implement automatic file restarting. These protocols provide recovery and restarting options for file transfers that were terminated for any reason before the whole file or batch is finished being transferred.

Related Topics

Restart Example, STARTSERVER, STOPSERVER, XPROT, File Transfer, Dialog Strings

SCRIPT HELPS RESTART FILE TRANSFER EXAMPLE

In most cases, TERM's file restarting is totally automatic if a configuration file is used that specifies valid LOGIN, LOGOUT, STARTSERVER, and STOPSERVER commands. For example:

```
RETRIES 2    ! 3 tries total
    SET LOG ON   ! auto logging
    iferror CONNECT remsys
      return
    endif
```

```
iferror STARTSERVER
      noerror HANGUP
      return
    endif


! perform file transfers here
    xfer sourcefile destdir[options]


noerror STOPSERVER
    noerror HANGUP
```

In this script, the RETRIES count is first set to 2, allowing a total of three attempts to transfer the file before reporting failure. Then, TERM's auto-logging is turned on, so a full audit trail of what happened can be found in the file TERM.LOG in the current directory.

Next, a connection is attempted to remsys. This statement is error checked so that the script doesn't abort if it can't make a connection. The CONNECT statement will dial the phone if a number is specified, and login if a LOGIN is specified.

Next, an attempt is made to start the remote TERM server. If this fails, the phone is hung up, which includes an auto-logout if a LOGOUT command was specified in the configuration file.

Next, a file transfer is attempted. TERM will try three times to transfer the file. TERM will automatically know to use the remsys configuration file, LOGIN command, STARTSERVER command, and the current XFER statement if the file transfer fails. If, after three attempts, the transfer still cannot be made, the XFER statement will return error. Otherwise, the remote server is stopped, the remsys logged out, if LOGOUT present, and the modem, if a number was specified, hung-up. These last statements are NOERRORed because it is still desirable to hangup the phone, even if the remote STOPSERVER command fails. File transfers will not be restarted if a user types a ^C to abort the transfer, even if RESTART is set. This allows interactive aborting of file transfers at the user's will.

Related Topics

RESTART, LOGIN, LOGOUT, STARTSERVER, STOPSERVER, XPROT, File Transfer, Dialog Strings

Script Command **STARTSERVER**

Purpose

Restartable file transfer control.

Script Usage

STARTSERVER *string*

Parameters

string Is a string expression

Description

The STARTSERVER, STOPSERVER, and DIALOG commands allow dialogues with a remote system by interpreting their passed string parameters.

STARTSERVER specifically contains the commands to be executed to start the remote server. This dialog string is executed after the connection has been established in a restart.

This command operate very much like the LOGIN command. TERM executes the dialog string automatically in the process of performing an automated file transfer restart. This command is placed in the configuration file for each specific remote connection, thereby allowing all remote data to remain in the configuration file.

Examples

STARTSERVER "^L03^T10-x^SOK^S^R00| ^Slogin:^S^R01|^SNO CARRIER^S^R01"

This example sets a maximum loop count of 3 and a string timeout wait of 10 seconds. It then sends a carriage return followed by term -x, carriage return, and waits for either OK, login:, or NO CARRIER.

If OKis seen, the command returns with an OK error status. If login: or NO CARRIER is seen, the command returns with an ERROR status.

Related Topics

RESTART, Restart Example, LOGIN, LOGOUT, STOPSERVER, XPROT, File Transfer, Dialog Strings

Script Command **STOPSERVER**

Purpose

Restartable file transfer control.

Script Usage

STOPSERVER *string*

Parameters

string Is a string expression

Description

The STARTSERVER, STOPSERVER, and DIALOG commands allow dialogues with a remote system by interpreting their passed string parameters.

STOPSERVER specifically contains the commands to be executed to terminate the remote server. This dialog string is executed after the transfer is successfully completed in a restarted file transfer.

This command operate very much like the LOGOUT command. TERM executes the dialog string automatically in the process of performing an automated file transfer restart. This command is placed in the configuration file for each specific remote connection, thereby allowing all remote data to remain in the configuration file.

Examples

STOPSERVER "^x^x^x^x^x"

The above sets the stopserver string to 5 CTRL+Xs. This is one method to stop the WTERMCRC server mode on a remote system.

Related Topics

RESTART, Restart Example, LOGIN, LOGOUT, STARTSERVER, XPROT, File Transfer, Dialog Strings

Script Command **RETRIES**

Purpose

To limit the number of bad packets before an abort in file transfers.

Script Usage

RETRIES *n*

Parameters

n Is a numeric expression for the maximum number of retries during an error-checked file transfer.

Description

This command limits the number of consecutive bad packets allowed during a TERM, XMODEM, YMODMEM or KERMIT file transfer. Defaults to 10.

Related Topics

RTIME, STIME, HTIME, ITIME, File Transfer Commands

Script Command **RTIME**

Purpose

To set packet receive timeout during file transfers.

Script Usage

RTIME *nsecs*

Parameters

nsecs Is a numeric expression specifying a number of seconds.

Description

This command sets packet receiver timeout. Defaults to 8. Setting RTIME to 0 will cause TERM to never timeout. With the WTERMCRC and ZMODEM protocols this time is set to 8.

Related Topics

RETRIES, STIME, HTIME, ITIME, File Transfer Commands

Script Command **STIME**

Purpose

To set the packet send timeout during file transfers.

Script Usage

STIME *nsecs*

Parameters

nsecs Is a numeric expression specifying a number of seconds.

Description

This command sets packet sender timeout. Defaults to 20. This number should always be greater than RTIME. Setting STIME to 0 will cause TERM to never timeout. With the WTERMCRC and ZMODEM protocols this time is always 60.

Related Topics

RETRIES, RTIME, HTIME, ITIME, File Transfer Commands

Script Command **HTIME**

Purpose

To set the handshake timeout for file transfers.

Script Usage

HTIME *nsecs*

Parameters

nsecs Is a numeric expression specifying a number of seconds.

Description

This command sets handshake timeout in seconds. This is the amount of time for TERM to wait before beginning a file transfer, after a GET, XFER, SEND or RCV command has been initiated. Defaults to 11. Setting HTIME to 0 will cause TERM never to timeout.

Related Topics

retries, rtime, stime, itime, xfer, get, send, rcvRETRIES, RTIME, STIME, ITIME, File Transfer Commands

Script Command **ITIME**

Purpose

To set the idle time during ASCII file transfers.

Script Usage

ITIME *nsecs*

Parameters

nsecs Is a numeric expression specifying a number of seconds.

Description

This command sets the idle time during ASCII file transfers, while waiting for an XON or ACK during XON and ETX protocols. Defaults to 0. Leaving ITIME at 0 will cause TERM to never timeout.

Related Topics

retries, rtime, stime, htime, xfer, get, send, rcvRETRIES, RTIME, STIME, HTIME, File Transfer Commands

Script Command **RETURN**

Purpose

To return from a script file or procedure.

Script Usage

RETURN [*retval*]

Parameters

retval Is an optional numeric parameter specifying analternate return code, stored in _errno, as shownin the following table:

Code	Result
0	OK (default, if no retval specified)
1	ERROR
2	ABORT
3	INTERRUPT
4-511	Reserved for future use
512-32767	User-defined abort codes (no error)

Description

This command returns control to the statement directly following the DO command that called the currently executing script file or procedure. If the DO command was given interactively or as a parameter to TERM when run from the operating system, TERM will return to terminal emulation mode. If the script being RETURNed from was called as a parameter to a -x mode TERM, TERM returns to -x mode and issues the OK prompt.

RETURN statements are allowed within IF/ENDIF and REPEAT/UNTIL loops, ON statements, CASE statements, DO files, and PROCedures.

Related Topics

PROC, DO, IF, REPEAT, QUIT, ON

Script Command **RMDIR**

Purpose

to remove a directory.

Script Usage

RMDIR *dir*

Parameters

dir Is the name of the directory. The directory must be empty.

Description

This command is used to remove a directory from within TERM's script language.

The directory name *dir* is not quoted. TERM reports an error if:

* The old directory does not exist.
* It is not empty.
* You don't have delete permission for it.

Examples

To remove the directory maketerm under the current directory:
RMDIR maketerm

Related Topics

RENAME, ERASE, COPY, MKDIR, CD, TYPE, RUN

Script Commands **RUN - RUNX**

Purpose

To run another program from within TERM.

Script Usage

RUN *command line*
RUNX *command line*

Parameters

command line Is a command line to send to the operating system. The*command line* should appear exactly as if typed directly to the system prompt; it is sent unmodified to the operating system.

Description

This command allows the user to execute an operating system command from TERM, and then return to TERM.

On DOS systems, the environment variable COMSPEC must correctly name the COMMAND.COM interpreter, and the PATH variable should be set to the directory where most programs live. For example:

COMSPEC=A:.COM
PATH=A:

DOS only. RUNX is identical to RUN, except that it will return the status of the executed command. If the command failed, _errno is set to 1. If the command executes, _errno is set to 0.

Examples

To run the UNIX ls program for a file directory:
RUN ls

Related Topics

REMOTE

Script Command **SCROLLBACK**

Purpose

To allocate the number of lines for the scrollback buffer.

Script Usage

SCROLLBACK *n*

Parameters

n	The number of lines to allocate for a scrollback buffer. Canbe 0 to 100. Default is 100. With the Extend Emulation option, themaximum number of scrollback lines is 60.

Description

This command defines the size of the scrollback buffer.

This command must be in the TERM.SYS file to take effect.

The scrollback buffer is viewed by using the following four keys:

Scroll Line Up Key
Pressing this key scrolls you back through the buffer, line by line. By default, CTRL UP is the Scroll Line Up key.

Scroll Line Down Key
Pressing this key scrolls you forward through the buffer, line by line. By default, CTRL DOWN is the Scroll Line Down key.

Scroll Page Up Key
Pressing this key scrolls you back through the buffer a page at a time. By default, CTRL PGUP is the Scroll Page Up key.

Scroll Page Down Key
Pressing this key scrolls you forward through the buffer a page at a time. By default, CTRL PGDN is the Scroll Page Down key.

Script Command **SESSION**

Purpose

To create, access and disconnect multiple sessions.

Script Usage

SESSION NEW [*service*]
SESSION GOTO *n*
SESSION NEXT
SESSION DISCONNECT

Parameters

service	This is the connection to be established in the new session.*n* This is the session number from one to five.

Description

DOS only. The SESSION command allows you to create script files which interact with each active session.

SESSION NEW
opens a new session and establishes a connection to the specified service.

SESSION GOTO
makes session *n* the active session. All activity will take place with this session.

SESSION NEXT
is similar to SESSION GOTO except that it makes the next session the active one.

SESSION DISCONNECT
breaks the connection in the currently active session.

Script Command **SET**

Purpose

To set various TERM options ON or OFF.

Script Usage

SET ABORT [ON or OFF]
SET ADDCR [ON or OFF]
SET ADDEOF [ON or OFF]
SET ADDLF [ON or OFF]
SET ALTCHK [ON or OFF]
SET ALTKEYS [ON or OFF]
SET BLOCKMODE [ON or OFF]
SET BSDEL [ON or OFF]
SET COMPRESS [ON or OFF]
SET CONTROL [ON or OFF]
SET DDTR [ON or OFF]
SET DESTBS [ON or OFF]
SET DIMDIM [ON or OFF]
SET DLGUNITS [ON or OFF]
SET DTR [ON or OFF]
SET ECHO [ON or OFF]
SET EOFOPEN [ON or OFF]
SET ERROR [ON or OFF]
SET ESC8BIT [ON or OFF]
SET ESCCTL [ON or OFF]
SET EXITDISC [ON or OFF]
SET EXITDTR [ON or OFF]
SET FILECONV [ON or OFF]
SET KERMECHO [ON or OFF]
SET LOG [ON or OFF]
SET MASKPAR [ON or OFF]
SET NOSCROLL [ON or OFF]
SET REVDIM [ON or OFF]
SET SCALING [ON or OFF]
SET TABEX [ON or OFF]
SET TOUPPER [ON or OFF]
SET RTS [ON or OFF]
SET VIEW [ON or OFF]
SET WRAP [ON or OFF]
SET XFERACK [ON or OFF]
SET XFERSTAT [ON or OFF]

Description

This command sets various options to an ON or OFF value. The SYSVAR() function will report the status of the various options. Each option is discussed in detail on a separate screen.

Related Topics

termbits(), sysvar(), ON, DIALER, LOGIN, Send, SETKEY

Script Command **SET ABORT**

Script Usage

SET ABORT [ON | OFF]

Description

This command controls whether or not scripts can be aborted by typing the ABORT character (normally a ^C). If ABORT is set OFF, scripts cannot be aborted from the keyboard. The ^S and ^Q processing for stopping and resuming screen output is also disabled.

Related Topics

ABORT

Script Command **SET ADDCR**

Script Usage

SET ADDCR [ON | OFF]

Description

This command replaces the changed ADDCR command in v6.2. When set to ON, TERM inserts a carriage return (0D hex) before each linefeed received and sent.

Script Command **SET ADDLF**

Script Usage

SET ADDLF [ON | OFF]

Description

This command replaces the changed ADDLF command in v6.2. When set to ON, TERM inserts a line feed (0A hex) after each carriage return received and sent.

Script Command **SET ADDEOF**

Script Usage

SET ADDEOF [ON | OFF]

Description

This command is implemented on DOS systems only. It causes a CTRL Z character to be appended to the end of any file received with the GET/RCV commands using the (t) (translate table) option. This ensures total text file compatibility with some DOS text editors.

Script Command **SET ALTCHK**

Script Usage

SET ALTCHK [ON | OFF]

Description

This command determines whether the KERMIT transfer protocol uses an alternate checksum method. This is to provide compatibility with old versions of KERMIT. The default is OFF.

Related Topics

File Transfer, XPROT

Script Command **SET ALTKEYS**

Script Usage

SET ALTKEYS [ON | OFF]

Description

This command determines whether the ALT+A through ALT+Z keys control TERM's menu, or revert to their possibly non-empty terminal emulation values. Setting OFF uses the ALT keys for the emulators.

Script Command **SET BLOCKMODE**

Script Usage

SET BLOCKMODE [ON | OFF]

Description

Defines whether to use blockmode in the selected emulation. Only valid with the Wyse 50, IBM3101, and Televideo 912, 925, and 950 emulations. If set to OFF, line mode is used. The default is OFF.

Script Command **SET BSDEL**

Script Usage

SET BSDEL [ON | OFF]

Description

Defines whether your Backspace key sends a delete or a backspace. If set to ON, the Backspace sends a delete. Valid for the VT220-7 and VT220-8 emulations only.

Script Command **SET COMPRESS**

Script Usage

SET COMPRESS [ON | OFF]

Description

COMPRESS This command is used to turn on or off TERM's data compression feature during TERMCRC file transfers. It is useful when talking to systems that cannot perform data compression due to memory constraints, or when it is not desired to use the (z) option. The SET COMPRESS ON command can be used to re-enable data compression.

Related Topics

File Transfer, XPROT (specifically TERMCRC and WTERMCRC)

Script Command **SET CONTROL**

Script Usage

SET CONTROL [ON | OFF]

Description

This command controls whether TERM can be operated remotely using the GET, XFER or REMOTE commands. Control mode is set ON by TERM's -c and -x options on the command line. Otherwise it defaults to OFF.

Script Command **SET DDTR**

Script Usage

SET DDTR [ON | OFF]

Description

This command is discussed under DIALER CONTROL.

Script Command **SET DESTBS**

Script Usage

SET DESTBS [ON | OFF]

Description

Determines whether backspace characters received are destructive; they erase. The default is OFF.

Script Command **SET DIMDIM**

Script Usage

SET DIMDIM [ON | OFF]

Description

Normal displays as Dim and Bold displays as Normal. The default is OFF.

Script Commands **SET DTR - RTS**

Related Topics

SET DTR [ON | OFF]
SET RTS [ON | OFF]

Description

These commands control the setting of the RS232 pins DTR (data terminal ready, pin 20), and RTS (request to send, pin 4). Most modems will not operate without pin 20 ON. When TERM is first executed, the default state for both DTR and RTS is ON. The STATUS command can be used to find the present state of these pins. See alsoDIALER. Due to hardware or operating system limitations, some machines may not be able to directly control these signals under program control.

Script Command **SET ECHO**

Script Usage

SET ECHO [ON | OFF]

Description

This command is used to display commands as they are executed from within a script file. This is useful for debugging purposes. Commands are displayed after $$ variable substitution is performed. See also DO and BATCH.

Script Command **SET EOFOPEN**

Script Usage

SET EOFOPEN [ON | OFF]

Description

Determines the method in which serial ports are opened. If set to ON, the alternative port open method is used. This is required for the AT&T 3B2 system. The default is OFF.

Script Command **SET ERROR**

Script Usage

SET ERROR [ON | OFF]

Related Topics

This command controls whether TERM script command errors will abort script file execution. By default, ERROR is ON. If set to OFF, errors are not fatal. The global variable _errno must be checked to determine a command's completion status.

Script Command **SET ESC8BIT**

Script Usage

SET ESC8BIT [ON | OFF]

Description

This command allows file transfers over 7-bit links, using only characters between hex 20-7e, and ^X, which are printable characters only. This option may be used at any time, and allows modems or transport links with nonstandard parity settings to be used.

Script Command **SET ESCCTL**

Script Usage

SET ESCCTL [ON | OFF]

Description

This command controls whether TERM will escape control characters during transmission. If this is set to ON, then TERM will not send any characters less than an ASCII space (except

^X, the WTERMCRC escape character). This option is not normally required, but should be set when using WTERMCRC protocol across links that do not support raw transmission of non-printable characters. WTERMCRC always escapes XON, XOFF, DEL and NUL characters. The default for ESCCTL is OFF.

Script Command **SET EXITDISC**

Script Usage

SET EXITDISC [ON | OFF]

Description

This command controls whether or not to exit TERM when file transfer connection breaks. The default is OFF.

Script Command **SET EXITDTR**

Script Usage

SET EXITDTR [ON | OFF]

Description

This command controls whether TERM leaves DTR on or OFF on exit from TERM. The default is OFF, which means TERM will drop the phone line connection when exiting. On UNIX systems, EXITDTR controls the setting of the termio HUPCL (hangup on close) bit.

Script Command **SET FILECONV**

Script Usage

SET FILECONV [ON | OFF]

Description

This command sets the file conversion parameter for text file Transfers. When on TERM will automatically convert text files to the receiving format. This usually strips or inserts carriage return characters.

Script Command **SET KERMECHO**

Script Usage

SET KERMECHO [ON | OFF]

Description

This command allows Kermit file transfers to proceed with systems that do not turn off echoing of received characters during kermit file transfers, such as some GCOS machines. If the Kermit file transfers are failing, try setting this option to ON.

Script Command **SET LOG**

Script Usage

SET LOG [ON | OFF]

Description

This command turns on or off TERM's automatic logging feature. All subsequent file transfer commands are logged to the current logfile, which is by default TERM.LOG in the current directory.

If logging is turned on, TERM will auto-log the start and completion status of all attempted file transfers. To turn logging on, use the SET LOG ON statement. The LOGFILE command can be used to set an alternative logfile name, other than TERM.LOG in the default directory.

The format of the logfile is as follows: All entries have the same format, starting with the date and time:
06/08 15:07:50 [logged message]

File transfer starts are logged by writing XFER or GET followed by the exact string used in the XFER or GET statement. The following shows a WTERMCRC file transfer of a file named foo to the current directory of the remote system:
06/08 15:07:54 XFER (WTERMCRC): foo .

Before the file transfer terminates, any error or informative messages are also logged to the logfile, such as this one, stating that TERM is automatically decreasing the packet size to 64 bytes/packet:
06/08 15:07:56 Decreasing packet size to 64

Upon termination, the file name, success status, percentage transferred, number of restarts/retries, and total number of bytes transferred/attempted is logged in the following format:

06/08 15:11:56 foo (25 secs 100% 0/0 retries 21848/21848 bytes)
06/08 15:11:56 Receive success

All errors are prefixed with the word ERROR: as follows:
06/11 21:39:42 ERROR: Transfer aborted by ^C

Script Command **SET MASKPAR**

Script Usage

SET MASKPAR [ON | OFF]

Description

This command is used to enable and disable masking of the parity bit for incoming and outgoing data during terminal emulations, data captures, WAIT/DWAIT, and CREAD commands. In previous versions, the PROTOCOL ASCII/BINARY command was used. This command is primarily intended for European support.

By default, the MASKPAR option is set ON, which means TERM disregards the parity bit on all received data. The PROTOCOL ASCII/BINARY command now determines only whether captured data should be converted to the receiving system's text file format, or not. The MASKPAR setting is set to ON when selecting the VT220-7 emulator, and OFF when selecting the VT220-8. Other emulations pass hi-bit characters through the emulator when MASKPAR is OFF. This is primarily useful on IBM PC versions of TERM, resulting in IBM PC character set access through characters 0x80-0xff.

Script Command **SET NOSCROLL**

Script Usage

SET NOSCROLL [ON | OFF]

Description

Controls whether the screen scrolls when a character is typed in the last column of the last line. If set to OFF, all lines will scroll up one line and the cursor will appear at the beginning of the last line. If set to ON, the cursor will move to the beginning of the first line. The default is OFF.

Script Command **SET REVDIM**

Script Usage

SET REVDIM [ON | OFF]

Description

This command sets the use of reverse video rather than dim for Wyse 50 escape sequences that specify dim. If OFF, TERM will use bold video for dim. The default for Wyse 50 terminals is ON.

Script Command **SET SCALING**

Script Usage

SET SCALING [ON | OFF]

Description

If set to ON, text within an emulation window will be scaled to the size of the window so that all text can be displayed with the window. This option scales TERM fonts only.

Script Command **SET TABEX**

Script Usage

SET TABEX [ON | OFF]

Description

This command controls whether tabs should be expanded to spaces when output to the screen during a normal TTY emulation (see the EMULATE command). By default, the TABEX flag is set from the pt TERMCAP variable, which determines whether or not your terminal can accept real tab characters.

Script Command **SET TOUPPER**

Script Usage

SET TOUPPER [ON | OFF]

Description

This command is used to create files with upper-case file names on UNIX systems when those files come from DOS systems. By default, TERM creates lowercase file names. However, some application programs (notably RM-COBOL) require uppercase file names. Once TOUPPER ON is specified, all files received using the GET/RCV commands will be created with uppercase file names. Uppercasing can be disabled by a subsequent SET TOUPPER OFF command.

Script Command **SET VIEW**

Script Usage

SET VIEW [ON | OFF]

Description

This command allows the user to display on the screen the contents of files sent with the ASCII and LINE protocols during their transfer. This is useful both for visual verification that the desired file is being sent, and for debugging purposes. The default is VIEW OFF, which

causes TERM to display dots (...) on the screen for every 128 bytes or line sent. See also the XPROT command.

The VIEW command also controls whether or not incoming and outgoing data is displayed during the auto-login and logout sequences. The VIEW option is set ON when executing TERM with the -e startup option.

Script Command **SET WRAP**

Script Usage

SET WRAP [ON | OFF]

Description

This command controls whether the emulation being used automatically linefeeds when a character is typed in the 80th column. ON will cause the cursor to move to the first column of the next line. OFF will cause the cursor to remain on the 80th column. This defaults to ON.

NOTE WRAP OFF is not a valid option for the SCO ANSI or ANSI emulations.

Script Command **SET XFERSTAT**

Script Usage

SET XFERSTAT [ON | OFF]

Description

This command allows you to switch the transfer status window on or off. When SET XFERSTAT is OFF, the file transfer display will show periods (....), one for each packet transferred, representing the file transfer progress. OFF is useful for scripts that must show file transfer status in TERM windows, as the ON setting requires a full screen.

SET XFERSTAT ON will display a full-screen file transfer status. The default is ON. The following are descriptions of each of the fields displayed during file transfers when XFERSTAT is ON.

Protocol.
This field displays the protocol in use during the file transfer. Values displayed are the same as those selectable using the XPROT TERM command.

File Name.
This field displays the name of the file currently being transferred. In the case of multiple files, this field changes as each new file begins transferring.

Bytes Transferred.
This field displays the number of bytes transferred so far, followed by the total number of bytes to be transferred. If the WTERMCRC or TERMCRC protocol is in use, the total number of bytes transferred is available during receives as well. This field is updated after every packet is sent/received.

Percent Transferred.
This field displays the amount of the file transferred, in percentage points. This field is updated after every packet is sent/received.

Transfer Time.
This field counts the number of integral seconds of real time elapsed since the file transfer started.

Effective Baud Rate.
This field displays the effective disk-to-disk file transfer speed currently in process. When data compression is in use, the value should exceed the modem

baud rate. It is calculated by dividing the bytes transferred by the elapsed transfer time.

Number of Packets.

This field displays the number of file data packets transferred thus far over the communications line.

Restarts/Retries.

This field displays both the number of times this transfer has been restarted, followed by the number of bad packets in the current file transfer.

Last Error.

This field displays the last critical error message from either the local or the remote system. Messages displayed here are deemed critical, and not merely information.

Last Message.

This field displays the last information message from the local or remote system. These messages are mostly useful for interactive progress, but are also logged, and can be used for debugging.

Script Command **SETCOLOR**

Purpose

To set TERM's internal color number.

Script Usage

SETCOLOR *colornum*, *newvalue*

Parameters

colornum	Is the current color number
newvalue	Is the new value to be used instead of the current color number

Description

This command is used to change the device-dependent output value of TERM's internal color number. All TERM script commands that use color numbers use TERM's internal color numbers, which are meant to specify the same colors on any system. The SETCOLOR command allows device-dependent colors to be mapped to TERM's internal color numbers.

Related Topics

Color Chart

Script Command **SETKEY**

Purpose

To change the actions of any keyboard keys.

Script Usage

SETKEY *key meaning*
SETKEY [RESET | RESETX]

Parameters

key	Is the key whose response is being changed.
meaning	Is the new action for the specified key. Allowed values are listed later. It may also be a quoted string specifying a macro expansion value to be expanded when that key is pressed. -- or -- @@(script command)
RESET	unmaps all macro settings and resets TERM to its default MENU, HELP, and etc. key values.
RESETX	unmaps all macro settings, but leaves TERM's MENU, HELP, and etc. as currently set.

Description

This command is used to change the settings of TERM's control keys, or to assign macro expansions to keyboard characters. It is used to change the key pressed to select the menu, help, and command modes, and also to redefine the key pressed for break,and single key login and logout. Meaning can be set to RESET to free that key from any TERM function. TERM's actions for the meaning keys are shown next:

Default **Function**	**Key**	**Description**

GOLD AF3	Send next function key typed	
CAPTURE	AF5	Toggle data capture ON/OFF
PRINT	AF6	Print Screen
BREAK	ALTB	Send Break
ABORT	"^C"	Abort operation
SCRLINEUP	CTRL+UP	Scroll back by line
SCRLINEDN	CTRL+DOWN	Scroll forward by line
SCRPAGEUP	CTRL+PGUP	Scroll back by page
SCRPAGEDN	CTRL+PGDN	Scroll forward by page
SCANMODE	ALT+T	SCANMODE toggle out
NEXTSESS	AF10	Go to next session

When this command is used to allow macro key expansioncapabilities, any single keypress can be made to look as if anarbitrary number of keystrokes were typed. In this usage, thestring of characters in meaning are substituted whenever key ispressed. These substituted characters are inserted as though they were typed from the keyboard.

Examples

To change your help key to function key #3:

SETKEY f3 HELP

To map the F10 key to quit TERM:

SETKEY f10 "@@quit"

To map the return key to send a carriage return and line feed:

SETKEY "" ""

Script Command **SETSYM**

Purpose

Set graphics characters

Script Usage

SETSYM *symbol#*, *newstrval* [,1]

Parameters

symbol#	Is the number of the graphics character to replace
newstrval	Is the graphics character to replace with
1	specifies that all replaced entries under UNIX will use the GS and GE prefix and suffix before outputting the specified character.

Description

This command allows specifying all of TERM's special graphics characters, which include VT100 and Wyse 50 line drawing characters, as well as TERM's window drawing characters. On MSDOS systems, these characters will not probably need to be changed. On UNIX systems, these characters are read from the GS, GE and Ga termcap entries. Some UNIX systems use terminfo rather than termcap, which makes adding a non-standard Ga entry impossible. TERM now reads the acsc terminfo entry, but SETSYM can now be used to set symbol character values. The following table describes TERM's internal symbol numbers.

Symbol#	**Description**
0-32	33 VT100 graphics characters
33-386	single line box draw chars

32-446 double line box draw chars

45-62 Wyse50 graphics chars
63-94 IBM PC charset, chars 0-31

95-223 IBM PC charset, chars 127-255

The six line box drawing characters are listed in the order Upper-Left Corner, Upper-Right Corner, Lower-Left Corner, Lower-Right Corner, Horizontal Bar, Vertical Bar. For example, the six single-line box draw characters would be

33 Upper-Left Corner

34 Upper-Right Corner

35 Lower-Left Corner

36 Lower-Right Corner

37 Horizontal Bar

38 Vertical Bar

Multiple symbol values can be specified in the newstrval. TERM will begin replacing at the symbol# entry. If the option 1 is used, all replaced entries under UNIX will use the GS and GE prefix and suffix before outputting the specified character.

Examples

To replace, for example, the standard upper-left corner symbol with an asterisk (*), you would enter the command

SETSYM 33, R*S

Script Command **SETVAR**

Purpose

Assign values to variables

Script Usage

SETVAR *variable expr*

Parameters

variable Is the variable to be created.

expr Is the value to set the variable to.

Description

This command creates a variable and sets its value. The variable type created is the same as the value of expr: string, logical, or numeric. See Expressions for help in creating expressions.

The MEMLIST command can be used to list values of variables created.

The DIMSTR, DIMINT or DIMLOG commands must be used to create indexed variables before SETVAR can assign values to them.

Examples

To set the string variable x to hello:

SETVAR x "hello"

To increment the numeric variable y by 1:

SETVAR y y+1

To set the logical variable guilty to false:

SETVAR guilty FALSE

To dimension a 10 element indexed string variable TEST and assign all values to NONE:

```
DIMSTR test 10
SETVAR j 0
REPEAT
  SETVAR test[j] "NONE"
  SETVAR j j+1
UNTIL j = 10
```

You can reset variables and, as long as they do not change between array and nonarray or nonarray and array, the type can be changed without any extra effort. For instance, the following is legal:

```
SETVAR j 1
SETVAR k 5
SETVAR j j+k
SETVAR k "The new value of j is ";
? k,j
```

Related Topics

DIMSTR, DISPLAY, MEMLIST, READ, CREAD, RELEASE

Script Commands **SWITCH - ENDSWITCH**

Purpose

To provide multiple conditional execution in script files

Script Usage

```
SWITCH
    CASE condition1
            .
            (commands to be executed only if condition 1 is true)
            .
    CASE conditionn
            .
            (commands to be executed only if condition n is true)
            .
     DEFAULT
            .
            (commands to be executed if no other conditions are true)
            .
ENDSWITCH
```

Parameters

condition	Is an expression that evaluates to a TRUE/FALSE value. See expressions for help with .

Description

This command allows multiple conditions to be tested in script files. Each CASE clause is tested until a condition is found to be true, or until a DEFAULT clause or an ENDSWITCH statement is found. The statements following the true condition are then executed until the next CASE, DEFAULT, or ENDSWITCH is encountered. Control skips to the statement following the ENDSWITCH.

The DEFAULT clause must be the last one in the SWITCH statement. Otherwise any CASE clauses following it will never be tested.

SWITCH statements can be nested to any level, with the following rules:

* Each SWITCH must have a corresponding ENDSWITCH.

* SWITCH-ENDSWITCH pairs must both be within the same script file or procedure.

Examples

To check a variable for one of three values:

```
SWITCH
CASE var='A'
  DISPLAY "Var is A"
CASE var='B'
  DISPLAY "Var is B"
CASE var='C'
  DISPLAY "Var is C"
DEFAULT
  DISPLAY "Var is not A, B or C"
ENDSWITCH
```

Related Topics

IF, REPEAT, FOR, PROC, DO, QUIT, RETURN

Script Command **SERVER**

Purpose

To invoke server mode for WTERMCRC and ZMODEM file transfers from a script.

Script Usage

SERVER

Description

Server is similar in function to the -1x and -2x command line options. The user on the remote computer may then initiate file transfers. Upon completion of a file transfer session, TERM continues script execution.

Examples

Term is available on the host computer, to local computer is running a generic ZMODEM package. The script on the host to allow ZMODEM file transfers would look like this:

```
XPROT ZMODEM
SERVER
QUIT
```

If the above commands were placed in a TERM script file called ZXFER.CMD the person calling the system could then execute TERM with the following command line:

```
TERM ZXFER
```

TERM would then come up, set transfer protocol to ZMODEM, wait in server mode for a file transfer dialog, and then exit back to the command line after the file transfer was completed.

Related Topics

XPROT

Script Commnad **TERMINAL**

Purpose

To enter terminal emulation mode from a script file

Script Usage

```
TERMINAL NOCARRIER
TERMINAL [match string] [,exitkey]
```

Parameters

match string The character string that must be received to terminate this command. Strings with special characters in them can be specified using escape characters.

exitkey An optional key name that, if pressed, will terminate this command.

Description

This script command causes TERM to enter terminal emulation mode during script file execution, until either data carrier is lost, a *match string* is received, or a specified keyboard character is pressed. All single key commands are active during terminal mode.

If the TERMINAL command is given without any parameters, TERM will remain in terminal mode until a CTRL E (^E) is typed, at which point the next command in the script file will execute. This is for backward compatibility with older TERM scripts.

Related Topics

MODE, EMULATE, PROTOCOL

Script Command **TERMID**

Purpose

To set the terminal response for DEC VT emulations.

Script Usage

TERMID *str*

Parameters

str the terminal identification string expected by a DEC host for the DEC VT terminal TERM is emulating.

Description

DEC terminals respond to a request from the host to identify terminal type. This command allows the user to set TERM's response the the DEC terminal type request.

Related Topics

EMULATE

Script Command **TIME**

Purpose

To estimate file transfer times.

Script Usage

TIME *filelist* [*options*]

Parameters

filelist Is a list of files to use in calculating the time estimation. They should be separated by spaces. Multiple files can also be specified using the asterisk (*), question mark (?), or brackets (or) wildcard characters in the file name.

options Indicates options used for filelist matching. Allowable options are described next.

Description

This command is used to gain an estimation of the amount of time a file transfer will take.

TERM will display lines similar to the following for the files specified:

Filename	Bytes	Records	Time to Transmit
autocom.cmd	4732	37	12 secs
backup.cmd	454	4	4 secs

TOTAL: 5186 bytes, 41 records, 16 secs at 9600 baud

This means that the files consist of a total of 5186 characters,and that it will take approximately 16 seconds to transfer themat the current baud rate.

The figures given by TIME do not take into account the compression factors provided by use of the TERMCRC and WTERMCRC protocols. Remember that you can save an average of 60% of your transmission time for text files, and 30-40% for object files by using TERMCRC or WTERMCRC.

Options

(x) eXclude. When this option is used with a file name, that file is not included in the time calculation. If wildcards are used in the file name, all files indicated are excluded from the estimate. See the (x) option description under the XFER/SEND command.

Related Topics

File Transfer Commands

Script Command **TRANS**

Purpose

To translate characters on received or transmitted data

Script Usage

TRANS [INPUT or OUTPUT] *source dest*
TRANS DISPLAY *source dest*
TRANS RESET

Parameters

source A string that specifies the range of characters to translate on data received or transmitted on the communications line. The string must be enclosed in quotes if spaces are used. The dash (-) character may be used to denote a range of characters (i.e. a-f denotes the string abcdef). See Specifying Strings for rules.

dest A string that specifies the characters to translate the*source* into. The *source* and *dest* must have the same length.

Description

This command is used to translate a specified set of characters received or transmitted on the communications line to different character values. The TRANS command operates on data displayed to the screen. Files moved by the XFER or GET commands are modified only if the (t) option is given with the XFER or GET command.

The user specifies whether to translate incoming data, outgoing data, or both. Typing trans input a-z A-Z will cause TERM to translate all received lower case letters to upper case. The translated characters will be displayed on the screen after translation. Typing trans output x00-0x7f x80-will set the high bit (8th bit) on all transmitted characters.

NOTE TERM first checks for IGNORE characters, then TRANSlatecharacters, then ADDCR/ADDLF. Thus, charactersspecified by the IGNORE command and also specified fortranslation will be ignored, not translated. However,characters TRANSlated to an IGNORE character will not beignored.

TRANS RESET resets all translations in effect.

The DISPLAY option translates incoming characters just before display, rather than just after being received, as in the existing INPUT case.

For example, if you enter the command

TRANS DISPLAY "a" "b"

an ESC a a will become an ESC a b. In this case, the ESC a is part of an emulator escape sequence; since the ESC and first a are not displayed, they are NOT translated with TRANS DISPLAY. Since the second a will be displayed, it is translated to a b. This is very useful for European character translations of 7 bit to 8 bit (or vice versa) when talking to equipment that has used portions of the 7 bit ASCII character set to represent 8 bit IBM PC characters, such as the a-umlaut.

Examples

To translate all lowercase characters to uppercase on both input and output, type:
TRANS input 'a-z' 'A-Z' output 'a-z' 'A-Z'

Related Topics

SET ADDCR, SET ADDLF, IGNORE, File Transfer Commands

Script Command **TYPE**

Purpose

To display and paginate a file on the screen.

Script Usage

TYPE *filename* [(b)]

Parameters

filename	Indicates the name of the file to display. Wildcard characters are not allowed in *filename*.
(b)	The (b) option (for binary display) may be appended to the *filename*. This displays the file without converting any line feeds, carriage returns, etc., and does not paginate the file on the screen. This qualifier can be useful if saved emulation sequences need to be replayed on the screen without modification.

Description

This command is used to display the contents of a text file on the screen. It is useful for previewing a script file prior to execution. The screen scroll is stopped every 23 lines so nothing is missed; type any character to resume the file display or the abort character (normally ^C) to quit displaying the file. If the file *filename* does not exist, TERM reports an error.

This command's debugging usage has been enhanced through use with the TDELAY command. Currently, when the (b) option is used with the TYPE command, the TYPEd file is displayed in binary mode. This is useful when redisplaying a captured terminal session to duplicate an error or debug an emulation problem. If TDELAY is set, the display will slow down in order to monitor what is happening. If the TDELAY is set to 30000, the TYPEd display pauses after each character is displayed and waits for a key to be pressed.

Related Topics

RUN, ERASE, RENAME, CD, MKDIR, RMDIR, COPY, TDELAY

Script Command **USER**

Purpose

To specify the username for use with ^U command character in DIALOG or LOGIN strings.

Script Usage

USER *str*

Parameters

str	the username to use to log into a remote system.

Description

This command is used in .TAP files to specify the username to use for the ^U character in DIALOG or LOGIN strings. If the *str* variable starts with an ^E, or was created using the AT GET ... ENCRYPT command, it will be automatically decrypted before being sent out the communications line.

Related Topics

encrypt(), LOGIN, Dialog Strings

Script Command **VERSION**

Purpose

Show the TERM version number

Script Usage

VERSION

Description

This command prints the TERM version number, along with the date compiled, serial number, and machine type.

Script Command **WHIDE**

Purpose

To hide a displayed window.

Script Usage

WHIDE *n*[, *n* [, *n*]]

Parameters

n	the number of the window to hide. Several windows may be hidden at once by adding them to the command separated by commas.

Related Topics

DLGOPEN

Script Command **WSHOW**

Purpose

To show a hidden window.

Script Usage

WSHOW *n*[, *n*[, *n*]]

Parameters

n	the number of the window to show. Several windows may be shown at once by adding them to the command separated by commas.

Related Topics

DLGOPEN, WSELECT

Script Command **WMOVE**

Purpose

To move a previously defined window.

Script Usage

WMOVE *n*,*y*,*x*,*z*

Parameters

n	the number of the window to move
y	the new row for the upper left corner of the window
x	the new column for the upper left corner of the window
z	the "level" for the window. -1 demote the window bottom level 0 keep window at the same level 1 promote the window top level

Description

WMOVE will move the window and display it if it was hidden.

TERM "stacks" windows on the screen, like sheets of paper. WMOVE moves the window on the screen, and may also move the window on the stack by changing its level to top or bottom. WMOVE does not make the moved window the current window for entry.

Related Topics

DLGOPEN, WSELECT, WSHOW

Script Command **WILDSIZE**

Purpose

To allocate memory for wildcard file name expansion

Script Usage

WILDSIZE *n*

Parameters

n	a numeric value representing the number of bytes allocated for memory

Description

This command allows the user to specify the amount of memory that will be allocated for wildcard file name expansion. This will affect the FOR, SEND, and XFER commands.

The default memory allocated for file name expansions is 3584 (14 x 256) bytes, sufficient for the standard limit of 256 file names, one byte per character in the filename. TERM's file name expansion routine uses only the number of bytes in the file name plus one for memory storage. Therefore the default 3584 byte allocation will allow more than 256 files to be expanded if the average file name length is less than 14. WILDSIZE can be used to either specify a smaller amount of memory if memory usage is important or specify a larger memory allocation so that more file names may be selected by the wildcard.

Examples

This will allocate sufficient memory to expand a wildcard into up to 500 names of up to 10 characters each.

WILDSIZE 5000

Script Command **WCLOSE**

Purpose

To close pop-up windows from script files.

Script Usage

WCLOSE *n*[,*n* [,*n*]]

Parameters

n	Is the window number, from 1 to 20.

Description

This command will close a window, restore the screen contents below the window and free memory taken by the window.

Related Topics

DLGOPEN, WSELECT, AT

Script Command **WSELECT**

Purpose

To select pop-up windows from script files.

Script Usage

WSELECT *n*

Parameters

n	Is the window number, from 1 to 20.

Description

This command redirects output back to another already open window. To direct output back to the main screen, use WSELECT 0.

The AT and CLS commands work well with windows; AT positions relative to the window upper left, and CLS clears only the contents of the currently WSELECTed window.

Related Topics

DLGOPEN, WCLOSE, AT

Script Commands **WRU - WRUCHAR**

Purpose

To define the who-are-you answerback string and character.

Script Usage

WRU *string*
WRUCHAR *charnum*

Parameters

string	Is the character string sent whenever a WRU inquire character is received. The WRU inquire character is bydefault an ASCII ENQ (hex 05). Strings with special characters in them can be specified using escape characters. If *string* is not specified, WRU response is turned off.
charnum	Is the ASCII value of the character to be used as the WRU inquire character. If the WRU *string* is set, it will be transmitted following receipt of *charnum*. The *charnum* is specified as an integer 0 - 255.

Description

The WRU command is used to make TERM send a sequence of characters out the communications line whenever it receives the wruchar. The wruchar defaults to an ASCII ENQ (default hex 05) character, unless changed by the WRUCHAR command. Some timesharing systems transmit a wruchar for auto-logon or verification purposes.

Related Topics

MODE, EMULATE, PROTOCOL

Script Command **WTITLE**

Purpose

To change the title of a previously defined window.

Script Usage

WTITLE *n*,*str*

Parameters

n	the number of the window to re-title
str	a string containing the title

Description

The WTITLE command changes the title of the window number *n*.

Window 0 is TERM's main window. Changing the title of window 0 changes the title displayed for TERM.

Related Topics

DLGOPEN

Script Command **XMIT**

Purpose

To transmit a character string out the commline.

Script Usage

XMIT *string*

Parameters

string	Is the character string to send out the communications line. Strings with special characters in them may be specified using escape characters. See Specifying Strings for rules.

Description

XMIT is used to send an arbitrary string of characters out the communications line. It is useful for creating your own auto-logon sequences in script files. EXMIT is used to send encrypted strings.

Examples

To send the words run mail followed by a carriage return down the commline, type:

XMIT 'run mail'

Related Topics

CDELAY, EXMIT, SEND

Script Command **XPROT**

Purpose

To set protocol for file transfer commands.

Script Usage

XPROT [TERMCRC, WTERMCRC, XMODEMCRC, KERMIT, COMPSRVB, XMODEM, YMODEM, ZMODEM, FTP, ASCII, LINE or EXTERNAL]

Parameters

ASCII

Send file as a stream of ASCII characters without error checking. equivalent to version 6.2 USEND and LSEND.

TERMCRC

Use TERM's CRC-checked protocol when transferring files with the XFER/SEND and GET/RCV commands. This protocol provides wildcard transfer with error checking, along with automatic data compression during transfers. It is the default protocol unless changed.

WTERMCRC

Use TERM's sliding window protocol. CRC-checked protocol when transferring files with the XFER and GET commands between TERM systems with version 6.1 or later. This protocol provides wildcard transfer with error checking, along with automatic data compression during transfers.

KERMIT

Use Kermit protocol when transferring files with the XFER/SEND and GET/RCV commands. This protocol is widely used for performing error-checked transfers with a variety of mainframes. Wildcard transfers are supported with this protocol. See the following examples for more information on Kermit file transfers.

COMPUSERVB

Use the Compuserve B protocol for transferring files to and from the CompuServe information service.

YMODEM

The Ymodem protocol provides full support of the Ymodem specification, which includes *.* (wildcard) batch file transfers, and 1024 byte packets.

XMODEM, XMODEM-CRC

(Xmodem Checksum) The Xmodem-CRC and Xmodem protocol selections do not support *.* wildcard file transfers. The Xmodem, Ymodem, and XmodemCRC protocols, by their nature, require binary data paths. This entails the use of equipment which does not require the data link escaping of binary characters in general, and the DC1 (XON - ASCII character 17 decimal) and DC3 (XOFF - ASCII character 19 decimal) in particular. The characters are sometimes called flow control characters.

When using a modem (generally an modem that operates over 1200 baud, such as the Telebit Trailblazer, Multitek 224E, Microcom SX, and so on) that supports flow control, care should be taken to turn flow control off. If the modem supports flow control, the modem manual should reference this and show how it may be disabled. If flow control cannot be disabled, a protocol other than Xmodem should be used.

It is important to note that some commercial services offering connections above 1200 baud have hardware that supports flow control. If this is the case, it is likely that another protocol must be selected.

A binary data path does not extend only to flow control. Some modems trap special characters as command characters, without allowing them to be sent. Particular examples are the older Novation Smartcat modems,

several Tandy DCM series modems, and several Anchor Automation Signalman series modems. This hardware is incompatible with Xmodem and another protocol or other hardware must be used.

Some Xmodem implementations cannot change port settings to correspond to an 8 bit data path. This can be due to hardware limitations, operating system limitations, or software deficiencies. Required port settings must include 8 data bits, no parity, and a lack of character interpretation or translation by intermediate software and hardware. In some cases, simply reestablishing the connection with different parameters is all that is required. In other cases, it may be necessary to use a different protocol.

Transfer with Xmodem is usually accomplished from TERM's main menu. It is necessary to know what kind of system you are talking to. If you are logged into a remote system or bulletin board, you must start the remote end first to insure that it has begun to send or receive the file already. This is because Xmodem does not support the transmission of server commands to the remote system.

ZMODEM

TERM's Zmodem protocol is the standard Zmodem sliding-window protocol. Similar to WTERMCRC, Zmodem can automatically adjust the size of its packets and window. See WTERMCRC for details on manually adjusting these options. It supports single and wildcard file transfers. FTP TERM supports the network file transfer protocol FTP. If you are connected over a network, you can take advantage of the extremely high speeds FTP supports. TERM will automatically connect to your network host, login, initiate the transfer and disconnect. You do not need to place the remote system in server mode.

EXTERNAL

Because there are many different transfer protocols in the computing world, TERM cannot possibly support them all. By supporting an external protocol, you can use TERM to transfer files using any protocol. In order for transfers to take place using an external protocol, you must setup the Send and Receive fields in Advanced Protocol Setup These fields should contain the command for starting the protocol program in send and receive modes.

TERM will execute the protocol program and pass the name of the file to it. Make sure that the program is in your PATH.

Description

This command is used to select file transfer protocol for the XFER/SEND and GET/RCV commands.

Related Topics

File Transfer Commands

Script Command **CONFIG**

Purpose

To Read and Write Configuration Files.

Script Usage

CONFIG NEW
CONFIG GET [*session*]
CONFIG SET
CONFIG READ *filename*
CONFIG WRITE *filename*

Parameters

session The number of an opened session.

filename The name of a configuration (.TAP) file to read from or write to. TERM automatically adds the .TAP extension, so it should not be specified in *filename*.

Description

Following is a description of what each of the CONFIG commands do:

CONFIG READ file Alternate <- file

CONFIG WRITE fileAlternate -> file

CONFIG GET Alternate <- Current

CONFIG GET *session* Alternate <- *session*

CONFIG SET Alternate -> Current

CONFIG NEW Alternate <- defaults

NOTE The arrows represent the flow of information.
-> means from Alternate workspace .
<- means to Alternate workspace.
TERM supports 2 configuration workspaces. The Current configuration provides the information for the active session. The Alternate configuration provides a workspace to view and change settings on another configuration without disturbing the active connection.

The CONFIG command manipulates the values in the Alternate configuration area. The provides a way to read, write and examine another session's settings without changing the current settings.

SYSVAR() provides access the current configuration settings in slots 1 - 22, and the alternate configuration settings in slots 41 - 62. The Alternate SYSVAR() settings replace the version 6.2 _c* variables.

Related Topics

CONNECT, sysvar()

Script Command **DDE ADVISE**

Purpose

To request an advise link on an item from the server application.

Script Usage

DDE ADVISE *chan*, *item*, *var*

Parameters

chan	an integer or variable for the channel number. This is not the DDE connection number. It is a number maintained by the programmer representing a DDE connection.
item	a quoted string or string variable containing the name of the item or variable requested.
var	a TERM script variable. The value of the item will be returned in this variable on each advise. If it is initialized to the type of the expected result TERM will attempt to convert the received text into the variable type. Otherwise a string will be returned in this variable.

Description

Requests an advise link on an item from the DDE server application. The DDE link must have been established using DDE INI prior to this command.

Related Topics

DDE INIT, DDE UNADVISE, DDE POKE

Script Command **DDE CLOSE**

Purpose

To terminate a DDE conversation.

Script Usage

DDE CLOSE *chan*

Parameters

chan	an integer or variable for the channel number. This is not the DDE connection number. It is a number maintained by the programmer representing a DDE connection.

Description

Closes a DDE conversation.

Script Command **DDE [ENABLE | DISABLE]**

Purpose

To control whether TERM will accept DDE conversation requests.

Script Usage

DDE ENABLE | DISABLE

Parameters

ENABLE	Enables DDE conversations. This is the default condition.
DISABLE	Refuse DDE conversations.

Description

ENABLES or DISABLES DDE messages. When disabled TERM refuses DDE connections. Issued by the server.

Script Command **DDE EXECUTE**

Purpose

To send a command to a DDE server.

Script Usage

DDE EXECUTE *chan*,*str*

Parameters

chan	an integer or variable for the channel number. This is not the DDE connection number. It is a number maintained by the programmer representing a DDE connection.
str	a quoted string or string variable containing a command for the server to execute. The documentation of the server application will describe the format required for the command. Usually the command is enclosed in square brackets ([]).

Description

Requests services of the DDE server.

Script Command **DDE INIT**

Purpose

To initialize a DDE conversation with a DDE server.

Script Usage

DDE INIT *chan*, *service*, *topic*

Parameters

chan	an integer or variable for the channel number. This is not the DDE connection number. It is a number maintained by the programmer representing a DDE connection.
service	a quoted string or string variable containing the name of the service to connect to. This may be found in the documentation of the application that will be the server.
topic	a quoted string or string variable containing the name of the topic to connect to. This may be found in the documentation of the application that will be the server.

Description

Initializes a DDE conversation. Issued by the client. The script will error if the connection is not made. Requesting a DDE conversation with an application that is not running on the Windows desktop will cause an error.

Script Command **DDE NAME**

Purpose

To change TERM's DDE server name.

Script Usage

DDE NAME *name*

Parameters

name	a quoted string or string variable containing the new TERM DDE server name.

Description

Changes TERM's server name from "WTERM" (the default) to *name*, for TERM's DDE server capabilities.

The DDE server name is used by DDE clients to initiate DDE conversations with TERM.

Script Command **DDE POKE**

Purpose

To send data to a DDE server.

Script Usage

DDE POKE *chan*,*item*,*str*

Parameters

chan	an integer or variable for the channel number. This is not the DDE connection number. It is a number maintained by the programmer representing a DDE connection.
item	a quoted string or string variable containing the name of the item or variable to assign the following value.
str	a quoted string or string variable containing the value to put into the item. TERM sends TEXT type for the POKE command.

Description

Sends data to the DDE server.

Related Topics

DDE REQUEST

Script Command **DDE REQUEST**

Purpose

To request data from the server of an open DDE conversation.

Script Usage

DDE REQUEST *chan*,*item*,*var*

Parameters

chan	an integer or variable for the channel number. This is not the DDE connection number. It is a number maintained by the programmer representing a DDE connection.
item	a quoted string or string variable containing the name of the item or variable requested.
var	a TERM script variable. The value of the item will be returned in this variable immediately. If it is initialized to the type of the expected result TERM will attempt to convert the received text into the variable type. Otherwise a string will be returned in this variable.

Description

Requests data from the DDE server application.

Related Topics

DDE POKE

Script Command **DDE TIMEOUT**

Purpose

To set the time TERM will wait for a reply to a DDE REQUEST, DDE ADVISE, or DDE EXECUTE before timing out on the DDE conversation.

Script Usage

DDE TIMEOUT *n*

Parameters

n	the number of milliseconds to wait for a reply to a DDE REQUEST before timing out The default value is 5000 milliseconds (5 seconds).

Description

Sets DDE timeout. Similar to receiver timeout in TERM's connection settings. Valid for TERM as a DDE client or server.

Related Topics

DDE REQUEST

Script Command **DDE UNADVISE**

Purpose

To close an advise link on an item on a DDE conversation.

Script Usage

DDE UNADVISE *chan*,*item*

Parameters

chan	an integer or variable for the channel number. This is not the DDE connection number. It is a number maintained by the programmer representing a DDE connection.
item	a quoted string or string variable containing the name of the item or variable requested.

Description

Closes the advise link on channel *chan* and *item*.

Related Topics

DDE ADVISE

DDE Server capabilities

TERM provides several topics and items for DDE conversations. For more information on accessing TERM's DDE services from another application, refer to the DDE client's documentation. The following sections give a brief outline of TERM's DDE server topics and functions.

server name:
WTERM[*n*]

n is the number of the instance. The first instance of TERM is not numbered. The brackets are to show that the number is not always there.

DDE Topics Provided:

System
Script
Terminal

DDE Server **System Topic**

Parameters

Items:	<Topics SysItems Formats Help>
Topics	'System Script Terminal'

SysItems	'Topics SysItems Formats Help'
Formats	'TEXT Link'
Help	'TERM Server Help...'
Execute:	Any script command

Description

Provides general information for DDE conversations.

DDE Server **Script Topic**

Parameters

Items:	<any string> (including TERM functions such as SYSVAR())
REQUEST	any TERM expression (right side of SETVAR)
ADVISE/UNADVISE	any single script variable
POKE	any script variable or variable lvalue (left side of SETVAR)
Execute:	Any script command

Description

Provides script variable access.

DDE Server **Terminal Topic**

Parameters

Items:	<any string>REQUEST XnnnYnnnWnnnHnnn (screen area descriptor)
ADVISE/UNADVISE	XnnnYnnnWnnnHnnn (screen area descriptor)
POKE	data sent down the comm line unfiltered
Execute:	Any script command

Description

Provides emulation screen and communications line access.

NOTE a screen area descriptor XnnnYnnnWnnnHnnn consists of the upper left corner X,Y valuea (XnnnYnnn) the Width (Wnnn) and the Height (Hnnn), where nnn represents a 3 digit number.

Script Command **DLGOPEN**

Purpose

To create dialog windows for data entry.

Script Usage

DLGOPEN *id*, *yloc*, *xloc*, *ylen*, *xlen* [,*fcol*, *bcol* [,*style* [, *title* [,*label*]]]]

Parameters

id	window number
yloc	upper left y coordinate. Upper left corner of the screen is 0,0.
xloc	upper left x coordinate.
ylen	internal vertical height of the dialog
xlen	internal horizontal width of the dialog.

fcol ignored

bcol ignored

style all styles below 255 are ignored. Values above 255 (256 and higher) makes the window a button bar which is inserted just below the Main Menu. For more information see the section on the Modal and Modeless Dialogs, and AT READ NOWAIT.

title The window title, displayed in the top of the frame of the dialog.

label ignored in windows.

NOTE The parameters *xloc*, *yloc*, *xlen*, *ylen* depend on the SET DLGUNITS command to determine the scheme of screen locations.

Description

The DLGOPEN creates a dialog window ready to receive AT statements to create an input dialog. The dialog is not displayed until an AT READ command is executed later.

All DISPLAY commands will be directed to window 0, which is the emulation window.

In character compatibility mode (SET DLGUNITS OFF) dialogs are created using text-based coordinates. TERM converts the text coordinates to dialog units. The text screen provides all characters of exactly the same size. Windows screens, on the other hand, provide proportional spacing and character widths. And that is why compatibility mode is not the preferred method of creating dialogs.

Window ids above 255 reserved for use by Century. Using a window id above 255 could conflict with TERM's user interface.

NOTE Windows dialogs do not support: ATTR, CLS, and AT x,y (for clear to end of line). These commands are ignored in Windows dialogs.

Related Topics

AT, SET DLGUNITS, WHIDE, WSHOW, WTITLE

Script Command **FONT**

Purpose

To set the font used by TERM for the emulation window.

Script Usage

FONT NORMAL, *fontname*, *charwidth*, *charheight*

Parameters

fontname is a string expression of the name of the font. The name of the font is the name displayed in the font dialog.

charwidth is the logical width of the font character set.

charheight is the logical height of the font character set.

Description

Sets the normal width (80 column screen) font for the emulation display window.

Resizing the window when SET SCALING is on will resize the font for window size. However, the font name remains the same.

Script Command **INISETFILE**

Purpose

To set the file name for subsequent .INI operations.

Script Usage

INISETFILE *filename*

Parameters

filename name of the .INI file to use for subsequent INISETSTR and inigetstr() operations.

Description

Sets the file name for subsequent .INI file operations.

Related Topics

INISETSTR, inigetstr()

Script Command **INISETSTR**

Purpose

To write values into Windows .INI formatted files.

Script Usage

INISETSTR *section*, *key*, *value*

Parameters

section name of the .INI file section to write to.

key name of the item to write.

value string representing a valid value for the key.

Description

Windows .INI files contain information in the following format:

```
[section title]
key=value
```

The inisetstr command searches for the section title and then the key. If found, it sets the value located to the right of the "=". Otherwise it creates the entry.

This function allows a script programmer to set values from .INI files.

Related Topics

INISETFILE, inigetstr()

Script Command **MSGBOX**

Purpose

To open a predefined message dialog widow with an icon and text, and to return the value of the button pressed to close the dialog.

Script Usage

MSGBOX *text*, *title*, *type*

Parameters

text text for the message window.

title title of the message window.

type one of the following values:

0 = ! icon with an Ok button. Used for displaying error andinformational messages.

1 = ? icon with Ok and Cancel buttons. Used for confirmingactions.

2 = ? icon with Yes and No buttons. Used for confirming actions.

3 = ? icon with Yes, No, and Cancel buttons. Used for confirmingactions where 3 options are desired.

Returns

_retval value equal to one less than the number of the buttonpressed. Numbering begins on the leftmost button. For example Ok/Yes return 0.

type 0
0 Ok

type 1 and 2
0 OK | Yes
1 Cancel | No

type 3
0 Yes
1 No
2 Cancel

Description

A standard call to display a message box on the screen. Becauseof the Windows-based functions of TERM, writing the the terminal screen should be avoided. Putting all messages to the user in message boxes will preserve the Windows look even in user written script programs.

Script Command **PASTE LINK**

Purpose

To open a client DDE link from the LINK format in the Windows clipboard, and write the results to the active communications link.

Script Usage

PASTE LINK

Description

Opens a client DDE link from the LINK data found in the Windows Clipboard. Paste Link then reads data from the DDE server application and copies it to the active communications line. TERM does not filter the data.

Related Topics

ispaste(), PASTE TEXT

Script Command **PASTE TEXT**

Purpose

To paste the TEXT contents of the clipboard to the active communications line.

Script Usage

PASTE TEXT

Description

Reads the Windows clipboard TEXT format and writes it to the active communications line. TERM does not filter the copied text.

Related Topics

ispaste(), PASTE LINK

Script Command **SCREEN COPY**

Purpose

To copy the text selected on the emulation window to the Windows clipboard.

Script Usage

SCREEN COPY

Description

SCREEN COPY places 2 items in the clipboard: 1) a TEXT format entry containing the selected text from the window; 2) a LINK format entry from which a DDE client may initiate an advise link with TERM on the selected window location that was selected.

The highlight created by dragging the mouse on the emulation window is returned to normal with this call.

Copy copies a LINK into the clipboard at each copy. This LINK may be used by a client DDE application's paste link to request data from TERM screen areas.

See the introduction section on DDE for more information on DDE links.

Related Topics

SCREEN NOSELECT, iscopy()

Script Command **SCREEN NOSELECT**

Purpose

To remove the highlight on the emulation window and deselect the text that was highlighted.

Script Usage

SCREEN NOSELECT

Description

Removes any highlighted text on the emulation screen.

Related Topics

SCREEN COPY, iscopy()

Script Command **SCREEN PRINT**

Purpose

To print the current contents of the emulation window to TERM's print system.

Script Usage

SCREEN PRINT

Description

Performs the same function as ALT+F6, or the Print Screen Key defined in the keyboard setup.

Script Command **SET DLGUNITS**

Purpose

To change coordinate scheme used for DLGOPEN and AT statements from character calculations to dialog units.

Script Usage

SET DLGUNITS [ON | OFF]

Parameters

ON	use Windows dialog units for DLGOPEN and AT.
OFF	use character units for DLGOPEN and AT.

Description

Character based screens, use letters which all take the same space on the screen. Windows uses proportional characters and character spacing, where each letter is of a different width. For example: in a character based system the letters "i" and "w" are the same width; in a proportional system the letter "i" may be 1/3 the width of the letter "w".

While SET DLGUNITS is OFF, TERM converts character (column and row) coordinates to dialog units. The conversion is an approximation, and the resulting dialogs sometimes do not look polished. However, DLGOPEN will produce functional entry dialogs from script files produced for character-based systems running TERM.

With SET DLGUNITS ON, TERM uses the Windows standard dialog units to create dialog windows.

Dialog units are a standard Windows measurement for creating dialogs and placing text and input fields in them. Each dialog unit is 1/4 character wide and 1/8 character high.

TERM uses the 8pt MS Sans Serif font for dialogs.

Script Command **STDFILE**

Purpose

To display a standard file dialog allowing the user to select afile its type and path.

Script Usage

STDFILE *strvar*, [*filename*[,*filter* [, *default_extension*[,*title* [,*type*]]]]]

Parameters

strvar	a string variable for the fully qualified path namereturned. Returns empty if the dialog was cancelled.
filename	the file name initially displayed in the File Name editfield.
filter	a quoted string or a string variable containing a textdescription of the type of file the filter displays in the File Name listbox, and the wildcard representative of the file type.The extension is delimited by the vertical bar (|). For example: "All Files (*.*)|*.*|". The text description is displayed in the "List Files of Type" list box, and the wildcard is used to searchfor files in the selected directory.
default_extension	the extension to be appended to the file name if no extension isspecified by the user in the File Name edit field. Do not includethe period (.).
title	the title of the file dialog box. Defaults to "Open File"

type Selects either:

0 = Ok and Cancel buttons.
1 = Connect, Load and Cancel Buttons, with a New Session checkbox. Used to open Configuration files.

Returns

strvar the fully qualified path name of the file selected,

_**retval** the button pressed.

type = 0

0 = Cancel button pressed
1 = Ok button pressed

type = 1

0 = Cancel = 0,
1 = Ok = 1,
1 = Load = 1,
2 = Connect,
9 = make a new session and load the configuration
10 = make a new session and connect to the configuration.

Description

Displays a standard open file dialog. The controls allow a userto "visually" select files and navigate directories.

Used for selecting a file that exists for manipulation.

Script Command **STDSAVE**

Purpose

To Display a standard save file dialog allowing the user to select a directory and file name for a file to be saved. The Save Configuration As STDSAVE follows:

Script Usage

STDSAVE *strvar* [,*filename*[,*default_extension*[,*title*]]]

Parameters

strvar a string variable for the fully qualified path name returned. Returns empty if the dialog was cancelled.

filename the file name initially displayed in the File Name edit field.

filter a quoted string or a string variable containing a text description of the type of file the filter displays in the File Name listbox, and the wildcard representative of the file type. The extension is delimited by the vertical bar (|). For example: "All Files (*.*)|*.*|". The text description is displayed in the "List Files of Type" list box, and the wildcard is used to search for files in the selected directory.

default_extension the extension to be appended to the file name if no extension is specified by the user in the File Name edit field. Do not include the period (.).

title the title of the file dialog box. Defaults to "Save File"

Returns

strvar the fully qualified path name of file selected,

_**retval** the button pressed. Cancel = 0, Ok = 1.

Description

Displays a standard Windows file save dialog. The controls allow a user to "visually" select files and navigate directories.

The default title is "Save File". Used for creating new files, and possibly selecting to overwrite and existing file. The programmer must check for an overwrite condition.

Script Command **CURSOR**

Purpose

To change the shape of the cursor.

Script Usage

CURSOR ARROW | WAIT

Parameters

ARROW	Changes the cursor to the arrow pointer.
WAIT	Changes the cursor to the hour glass to show that the system is busy.

Description

The CURSOR command changes the shape of the cursor. It is professional to display the hour glass cursor during procedures that execute for some time without changing the display.

Script Function **ASC()**

Purpose

To convert a character its ASCII value.

Script Usage

SETVAR *n* ASC(*str*)

Returns

an integer (range 1..255) representing the ASCII value of the first character of a string.
- or -
0 for a NULL string.

Parameters

str string

Related Topics

chr(), chname()

Script Function **ATOD()**

Purpose

To convert a date/time string to an integer value.

Script Usage

SETVAR *n* ATOD(*date*)

Returns

an integer date number

Parameters

date date/time string of format MM/DD/YY.HH:MM:SS

Description

Convert date/time string of MM/DD/YY.HH:MM:SS format To an integer date value.

Related Topics

cdate(), ctime()

Script Function **EVAL()**

Purpose

To evaluate a command and return result code.

Script Usage

SETVAR *n* EVAL(*cmd*)

Returns

result code of executing the command.

Parameters

cmd string containing a command and its parameters to execute.

Related Topics

NOERROR, EXECUTE

Script Function **FILEDATE()**

Purpose

To return the integer date of file.

Script Usage

SETVAR *n* FILEDATE(*file*)

Returns

the integer value representing the date stamp of the file.

Parameters

file string containing the name of the file.

Related Topics

filemode(), filesize()

Script Function **FILEMODE()**

Purpose

To return integer permission bits of file.

Script Usage

SETVAR *n* FILEMODE(*file*)

Returns

integer representing the permission bits of a file.

Parameters

file string containing the name of the file.

Related Topics

filedate(), filesize()

Script Function **FILESIZE()**

Purpose

To return the size of file.

Script Usage

SETVAR *n* FILESIZE(*file*)

Returns

the size of the file in bytes.

Parameters

file string containing the name of the file.

Related Topics

filedate(), filemode()

Script Function **FTELL()**

Purpose

To return current position of FOPENed file

Script Usage

SETVAR *n* FTELL(*file*)

Returns

current position, from the beginning of the file.

Parameters

file name of a file previously FOPENed.

Related Topics

FOPEN, feof(), FSEEK

Script Function **LEN()**

Purpose

To return length of a string.

Script Usage

SETVAR *n* LEN(*str*)

Returns

number of characters in the string.

Parameters

str a string.

Related Topics

pos(), left(), right(), midstr(), trim()

Script Function **TERMBITS()**

Bit **Meaning**

Purpose

To return value of TERM internal settings.

Script Usage

SETVAR *n* TERMBITS()

Returns

the 32 bit number representing the setting of several TERM functions.

Description

To test for the value of SET ECHO: SETVAR x termbits()&0x100

If x equals zero, the setting is OFF. All other values indicate the setting is ON.

Bit	**Meaning**
0x00000001	If actually masking parity
0x00000002	SET MASKPAR
0x00000004	SET ABORT
0x00000008	SET ADDEOF
0x00000010	SET COMPRESS
0x00000020	SET CONTROL
0x00000040	SET DESTBS
0x00000080	SET DDTR
0x00000100	SET ECHO
0x00000200	SET ERROR
0x00000400	SET NOSCROLL
0x00000800	SET TABEX
0x00001000	SET TOUPPER
0x00002000	SET VIEW
0x00004000	SET REVDIM
0x00008000	SET LOG
0x00010000	SET XFERSTAT
0x00020000	SET EXITDTR
0x00040000	SET EOFOPEN
0x00080000	SET CONSXON
0x00100000	SET KERMECHO
0x00200000	SET ALTCHK
0x00400000	SET ESCCTL
0x00800000	SET ESC8BIT
0x01000000	SET WRAP
0x02000000	Setting of -x flag
0x04000000	SET BLOCKMODE
0x08000000	SET BSDEL
0x10000000	SET EXITDISC
0x20000000	Setting of -y flag

Related Topics

SET

Script Function **OPSYS()**

Purpose

To return operating system.

Script Usage

SETVAR *n* OPSYS()

Returns

0 = VMS
1 = MS-DOS
2 = XENIX
3 = UNIX System III or V
4 = Berkeley UNIX
8 = BTOS
11 = SuperDOS

Related Topics

getenv(), gethomedir(), hascolor()

Script Function **POS()**

Purpose

To return position of string 1 within string 2.

Script Usage

SETVAR *n* POS(*str1*,*str2*,*start*)

Returns

position of string 1 within string 2 or 0 if not found.

Parameters

str1 substring to be searched for.

str2 string to be searched.

start location to begin searching in string 2.

Related Topics

len(), left(), right(), midstr(), trim(), tolower(), toupper()

Script Function **SUM()**

Purpose

To return the crc/chksum of a string.

Script Usage

SETVAR *n* SUM(*str*,*type*,*startval*)

Returns

crc/checksum of *str*.

Parameters

str string for which to generate the checksum.

startval initial value for the checksum.

type 0 = additive checksum
1 = xor checksum
2 = ccitt-16 crc

Script Function **TIME()**

Purpose

To get the system time.

Script Usage

SETVAR *n* TIME()

Returns

An integer of the number of seconds since January 1, 1970 GMT.

Related Topics

ctime(), cdate()

Script Function **VAL()**

Purpose

To return integer value of string.

Script Usage

SETVAR *n* VAL(*str*)

Returns

an integer representing the numeric value of the string.

Parameters

str a string containing numeric characters to be converted to a number.

Related Topics

str(), chr()

Script Function **MDMSTAT()**

Purpose

To return serial port status bits.

Script Usage

SETVAR *n* MDMSTAT()

Returns

serial port DCD, CTS, DTR, and RTS input status.
DCD = 01h
CTS = 02h
DTR = 04h
RTS = 08h

Related Topics

MODEM

Script Function **CDATE()**

Purpose

To convert TIME() return value into a string containing date.

Script Usage

SETVAR *n* CDATE(*time*)

Returns

ASCII string value of passed time as a string of format MM/DD/YY.

Parameters

time integer return value from TIME() function.

Related Topics

ctime(), time(), atod()

Script Function **CTIME()**

Purpose

To convert TIME() return value to a string containing time.

Script Usage

SETVAR *s* CTIME(*time*)

Returns

ASCII string value of passed time as a string of format HH:MM:SS.

Parameters

time integer return value from TIME() function.

Related Topics

cdate(), time(), atod()

Script Function **CHR()**

Purpose

convert an integer to a single ASCII character.

Script Usage

SETVAR *c* CHR(*number*)

Returns

string containing the corresponding single ASCII character.

Parameters

number an integer, values above 255 may not be defined.

Related Topics

val(), str()

Script Function **CHRDY()**

Purpose

To return a character pressed at the keyboard.

Script Usage

SETVAR *c* CHRDY()

Returns

character pressed at keyboard as string, without waiting. If no key was hit, return zero-length string.

Related Topics

chrin(), comin()

Script Function **CHRIN()**

Purpose

To wait for a character to be pressed at the keyboard and return it as a string.

Script Usage

SETVAR *c* CHRIN( )

Returns

character pressed at keyboard.

Related Topics

chrdy(), comin()

Script Function **COMIN()**

Purpose

To wait for character to be received at communications line and return it as a string.

Script Usage

SETVAR *c* COMIN()

Returns

character received at communications line.

Related Topics

comst(), chrin(), chrdy()

Script Function **GETENV()**

Purpose

To get the value of an environment variable.

Script Usage

SETVAR *s* GETENV(*var*)

Returns

value of environment variable as string or a zero-length string if the variable is not set.

Parameters

var string name of environment variable.

Related Topics

opsys(), gethomedir()

Script Function **GETHOMEDIR()**

Purpose

To get the home directory of the user currently executing TERM.

Script Usage

SETVAR *s* GETHOMEDIR()

Returns

the home directory of the user currently executing TERM.

Description

MS-DOS systems return "/".

If the HOME environment variable is set, its value is returned.

Related Topics

getenv(), opsys()

Script Function **LEFT()**

Purpose

To return leftmost portion of string.

Script Usage

SETVAR *s* LEFT(*str*,*len*)

Returns

leftmost *len* characters of string *str*.

Parameters

str string.

len integer length of string to return.

Related Topics

right(), midstr(), pos(), len()

Script Function **MIDSTR()**

Purpose

To return a substring of *str*.

Script Usage

SETVAR *s* MIDSTR(*str*,*pos*,*len*)

Returns

substring of string *str*, starting at position *pos* (1=first position), of length *len*.

Parameters

str string.

pos integer, beginning location of string to extract.

len integer, length of the string to extract.

Related Topics

right(), left(), pos(), len()

Script Function **RIGHT()**

Purpose

To return the rightmost *len* characters of *str*.

Script Usage

SETVAR *s* RIGHT(*str*,*len*)

Returns

rightmost *len* portion of string *str*.

Parameters

str string.

len integer, length of string to return.

Related Topics

left(), midstr(), pos(), len()

Script Function **STR()**

Purpose

To convert an integer to a string.

Script Usage

SETVAR *s* STR(*number*)

Returns

the string representation of the number, with no leading spaces.

Parameters

*number*integer.

Related Topics

val(), chr()

Script Function **TOLOWER()**

Purpose

To convert a string to all lower case.

Script Usage

SETVAR *s* TOLOWER(*str*)

Returns

str converted to all lower case.

Parameters

str string.

Related Topics

midstr(), pos(), toupper()

Script Function **TOUPPER()**

Purpose

To convert a string to all upper case.

Script Usage

SETVAR *s* TOUPPER(*str*)

Returns

str converted to all upper case.

Parameters

str string.

Related Topics

tolower, midstr(), pos()

Script Function **TRIM()**

Purpose

To trim all spaces of the right hand side of a string.

Script Usage

SETVAR *s* TRIM(*str*)

Returns

str with no trailing spaces.

Parameters

str string.

Related Topics

left(), right(), midstr(), pos(), len()

Script Function **UNIQ()**

Purpose

To create a unique file name from a template.

Script Usage

SETVAR *s* UNIQ(*str*)

Returns

a unique filename built from the template passed in *str*.

Parameters

str string to be used as template for a file name. Number sign (#) characters in the template are replaced by digits 0-9 until a unique non-existing filename results.

Description

This function performs 2 actions: 1) creates a file on the disk, 2) returns its name. In mult-user systems it is important to create the file as well as determine a unique name, because another user could "steal" the file name from you if there is any processing time between making a uniquename and creating it.

As a result of this, you must use FOPEN with the name created by the uniq() function instead of FCREATE. FCREATE will return an error used with the name returned from the uniq() function.

Related Topics

FOPEN

Script Function **COMST()**

Purpose

To test if a character has been received at the communications line.

Script Usage

```
IF COMST()
(do something)
ENDIF
```

Returns

TRUE if a character is received at the communications line.

FALSE if no character is received.

Related Topics

comin(), chrin()

Script Function **EXISTS()**

Purpose

To test for the existence of a file.

Script Usage

IF EXISTS(*file*)
(do something)
ENDIF

Returns

TRUE if the specified file exists.

Parameters

file string containing the file name to be searched for, you may also include path name.

Related Topics

FOPEN, uniq()

Script Function **FEOF()**

Purpose

To test for end-of-file on a file that has been FOPENed.

Script Usage

IF FEOF(*fileno*)
(show error and do something)
ENDIF

Returns

TRUE if the FOPENed file is at end of file.

Parameters

fileno the number assigned to a file when it was FOPENed.

Related Topics

FOPEN, FSEEK, ftell()

Script Function **HASCOLOR()**

Purpose

To determine if the system running TERM supports color.

Script Usage

IF HASCOLOR()
(do something requiring a color screen)

ELSE
(do something in black and white)
ENDIF

Returns

TRUE if operating on a system with a color monitor.

MS-DOS systems will return true only if the color monitor is active.

Related Topics

opsys()

Script Function **ISDIR()**

Purpose

To test if a filename is actually a directory.

Script Usage

IF ISDIR(*file*)
(do something requiring a directory)
ELSE
(do something requiring a file)
ENDIF

Returns

TRUE if the specified file is a directory.

Parameters

file string containing the name of a file or directory. Full path names are accepted.

Related Topics

FOPEN, exists()

Script Function **KEYIN**

Purpose

To access the TERM key identification number for a key press.

Script Usage

SETVAR *n* KEYIN()

Returns

An integer representing the TERM key identification number corresponding to the key pressed.

Description

This function waits for a key to be pressed, and then returns the TERM key identification number for it. This value could then be passed to keyname() to find the TERM key name.

Examples

```
? keyin()    ! press the letter a
97
? keyin()    ! press CTRL+F1
349
? keyname(97)
a
? keyname(349)
CF1
b ? keyname(keyin())    ! press ALT+F2
AF2
```

Related Topics

keyid(), keyname()

Script Function **TERMKEY()**

Purpose

To access the key identification number for keys assigned to TERM functions.

Script Usage

SETVAR *n* TERMKEY(*keynum*)

Parameters

keynum one of the following values:

0 = ABORT
2 = HELP
3 = GOLD
5 = CAPTURE
6 = PRINT
7 = COMPOSE
8 = NEXTSESS
11 = SCANMODE
12 = BREAK
13 = SCRLINEUP
14 = SCRLINEDN
15 = SCRPAGEUP
16 = SCRPAGEDN

Returns

An integer corresponding to the TERM internal value of the key mapped to the TERM function.

-1 if the function has not been assigned to a key.

Description

This function allows the the user to query the values for each of TERM's functions, such as MENU, HELP, COMPOSE, and others listed.

Examples

```
? termkey(1)
369
```

```
? keyname(termkey(1))
AF1
```

Related Topics

keyname(), keyid()

Script Function **KEYID()**

key id # keyname(*n*) chname(*n*,0) chname(*n*,1) chname(*n*,2)

Purpose

To access the TERM key identification number for a key name.

Script Usage

SETVAR *n* KEYID(*str*)

Parameters

str a string naming a key.

Returns

An integer representing the TERM key identification number associate with the named key.

Description

The TERM key identification number uniquely identifies each key stroke. This means that KEYID() will return the same identification number independent of the operating system or hardware platform.

For example, the function key F1 on MS-DOS machines may return a value such as "ESC+N", while a Wyse 50 sends a "^A" to the host running TERM. Using TERM script, all F1 key presses would return the value 256 through the KEYIN() function, and be comparable through the KEYID() function.

Following is the list of variable types id numbers and value names:

key id #	keyname(*n*)	chname(*n*,0)	chname(*n*,1)	chname(*n*,2)
0	^@	<NUL>	^@	^@
1	^A	<SOH>	^A	^A
2	^B	<STX>	^B	^B
3	^C	<ETX>	^C	^C
4	^D	<EOT>	^D	^D
5	^E	<ENQ>	^E	^E
6	^F	<ACK>	^F	^F
7	^G	<BEL>	^G	^G
8	^H	<BS>	^H	^H
9	TAB	<TAB>	^I	TAB
10	^J	<LF>	^J	^J
11	^K	<VT>	^K	^K
12	^L	<FF>	^L	^L
13	ENTER	<CR>	^M	ENTER
14	^N	<SO>	^N	^N
15	^O	<SI>	^O	^O
16	^P	<DLE>	^P	^P
17	^Q	<XON>	^Q	^Q
18	^R	<DC2>	^R	^R
19	^S	<XOF>	^S	^S
20	^T	<DC4>	^T	^T
21	^U	<NAK>	^U	^U
22	^V	<SYN>	^V	^V
23	^W	<ETB>	^W	^W
24	^X	<CAN>	^X	^X
25	^Y	<EOM>	^Y	^Y

26	^Z	<SUB>	^Z	^Z
27	ESC	<ESC>	^[	ESC
28	^\	<FS>	^\	^\
29	^]	<GS>	^]	^]
30	^^	<RS>	^^	^^
31	^_	<US>	^_	^_
32				
33	!	!	!	!
34	"	"	"	"
35	#	#	#	#
36	$	$	$	$
37	%	%	%	%
38	&&	&&	&&	&&
39	'	'	'	'
40	(	(	(	(
41	)	)	)	)
42	*	*	*	*
43	+	+	+	+
44	,	,	,	,
45	-	-	-	-
46	.	.	.	.
47	/	/	/	/
48	0	0	0	0
49	1	1	1	1
50	2	2	2	2
51	3	3	3	3
52	4	4	4	4
53	5	5	5	5
54	6	6	6	6
55	7	7	7	7
56	8	8	8	8
57	9	9	9	9
58	:	:	:	:
59	;	;	;	;
60	<	<	<	<
61	=	=	=	=
62	>	>	>	>
63	?	?	?	?
64	@	@	@	@
65	A	A	A	A
66	B	B	B	B
67	C	C	C	C
68	D	D	D	D
69	E	E	E	E
70	F	F	F	F
71	G	G	G	G
72	H	H	H	H
73	I	I	I	I
74	J	J	J	J
75	K	K	K	K
76	L	L	L	L
77	M	M	M	M
78	N	N	N	N
79	O	O	O	O
80	P	P	P	P
81	Q	Q	Q	Q
82	R	R	R	R
83	S	S	S	S
84	T	T	T	T
85	U	U	U	U
86	V	V	V	V

87	W	W	W	W
88	X	X	X	X
89	Y	Y	Y	Y
90	Z	Z	Z	Z
91	[	[	[	[
92	\	\	\	\
93	]	]	]	]
94	^	^	^	^
95				
96	‾	‾	‾	‾
97	a	a	a	a
98	b	b	b	b
99	c	c	c	c
100	d	d	d	d
101	e	e	e	e
102	f	f	f	f
103	g	g	g	g
104	h	h	h	h
105	i	i	i	i
106	j	j	j	j
107	k	k	k	k
108	l	l	l	l
109	m	m	m	m
110	n	n	n	n
111	o	o	o	o
112	p	p	p	p
113	q	q	q	q
114	r	r	r	r
115	s	s	s	s
116	t	t	t	t
117	u	u	u	u
118	v	v	v	v
119	w	w	w	w
120	x	x	x	x
121	y	y	y	y
122	z	z	z	z
123				
124	|	|	|	|
125				
126	~	~	~	~
127	^?	<DEL>	^?	^?
128		+<NUL>	+^@	
129		+<SOH>	+^A	
130		+<STX	>+^B	
131		+<ETX>	+^C	
132		+<EOT>	+^D	
133		+<ENQ>	+^E	
134		+<ACK>	+^F	
135		+<BEL>	+^G	
136		+<BS>	+^H	
137		+<TAB>	+^I	
138		+<LF>	+^J	
139		+<VT>	+^K	
140		+<FF>	+^L	
141		+<CR>	+^M	
142		+<SO>	+^N	
143		+<SI>	+^O	
144		+<DLE>	+^P	
145		+<XON>	+^Q	
146		+<DC2>	+^R	
147		+<XOF>	+^S	

148		+<DC4>	+^T	
149		+<NAK>	+^U	
150		+<SYN>	+^V	
151		+<ETB>	+^W	
152		+<CAN>	+^X	
153		+<EOM>	+^Y	
154		+<SUB>	+^Z	
155		+<ESC>	+^[	
156		+<FS>	+^\	
157		+<GS>	+^]	
158		+<RS>	+^^	
159		+<US>	+^_	
160		+	+	
161	!	+!	+!	!
162	"	+"	+"	"
163	#	+#	+#	#
164	$	+$$	+$$	$
165	%	+%	+%	%
166	&	+&	+&	&
167	'	+'	+'	'
168	(	+(	+(	(
169	)	+)	+)	)
170	*	+*	+*	*
171	+	++	++	+
172	,	+,	+,	,
173	-	+-	+-	-
174	.	+.	+.	.
175	/	+/	+/	/
176	0	+0	+0	0
177	1	+1	+1	1
178	2	+2	+2	2
179	3	+3	+3	3
180	4	+4	+4	4
181	5	+5	+5	5
182	6	+6	+6	6
183	7	+7	+7	7
184	8	+8	+8	8
185	9	+9	+9	9
186	:	+:	+:	:
187	;	+;	+;	;
188	<	+<	+<	<
189	=	+=	+=	=
190	>	+>	+>	>
191	?	+?	+?	?
192	@	+@	+@	@
193	A	+A	+A	A
194	B	+B	+B	B
195	C	+C	+C	C
196	D	+D	+D	D
197	E	+E	+E	E
198	F	+F	+F	F
199	G	+G	+G	G
200	H	+H	+H	H
201	I	+I	+I	I
202	J	+J	+J	J
203	K	+K	+K	K
204	L	+L	+L	L
205	M	+M	+M	M
206	N	+N	+N	N
207	O	+O	+O	O
208	P	+P	+P	P

209	Q	+Q	+Q	Q
210	R	+R	+R	R
211	S	+S	+S	S
212	T	+T	+T	T
213	U	+U	+U	U
214	V	+V	+V	V
215	W	+W	+W	W
216	X	+X	+X	X
217	Y	+Y	+Y	Y
218	Z	+Z	+Z	Z
219	[	+[	+[	[
220	\	+\	+\	\
221	]	+]	+]	]
222	^	+^	+^	^
223	_	+_	+_	_
224	`	+`	+`	`
225	a	+a	+a	a
226	b	+b	+b	b
227	c	+c	+c	c
228	d	+d	+d	d
229	e	+e	+e	e
230	f	+f	+f	f
231	g	+g	+g	g
232	h	+h	+h	h
233	i	+i	+i	i
234	j	+j	+j	j
235	k	+k	+k	k
236	l	+l	+l	l
237	m	+m	+m	m
238	n	+n	+n	n
239	o	+o	+o	o
240	p	+p	+p	p
241	q	+q	+q	q
242	r	+r	+r	r
243	s	+s	+s	s
244	t	+t	+t	t
245	u	+u	+u	u
246	v	+v	+v	v
247	w	+w	+w	w
248	x	+x	+x	x
249	y	+y	+y	y
250	z	+z	+z	z
251	{	+{	+{	{
252	|	+|	+|	|
253	}	+}	+}	}
254	~	+~	+~	~
255	•	+<DEL>	+^?	•
256	F1	[invalid]	[invalid]	F1
257	F2	[invalid]	[invalid]	F2
258	F3	[invalid]	[invalid]	F3
259	F4	[invalid]	[invalid]	F4
260	F5	[invalid]	[invalid]	F5
261	F6	[invalid]	[invalid]	F6
262	F7	[invalid]	[invalid]	F7
263	F8	[invalid]	[invalid]	F8
264	F9	[invalid]	[invalid]	F9
265	F10	[invalid]	[invalid]	F10
266	F11	[invalid]	[invalid]	F11
267	F12	[invalid]	[invalid]	F12
268	F13	[invalid]	[invalid]	F13
269	F14	[invalid]	[invalid]	F14

270	F15	[invalid]	[invalid]	F15
271	F16	[invalid]	[invalid]	F16
272	F17	[invalid]	[invalid]	F17
273	F18	[invalid]	[invalid]	F18
274	F19	[invalid]	[invalid]	F19
275	F20	[invalid]	[invalid]	F20
276	UP	[invalid]	[invalid]	UP
277	DOWN	[invalid]	[invalid]	DOWN
278	RIGHT	[invalid]	[invalid]	RIGHT
279	LEFT	[invalid]	[invalid]	LEFT
280	HOME	[invalid]	[invalid]	HOME
281	KP0	[invalid]	[invalid]	KP0
282	KP1	[invalid]	[invalid]	KP1
283	KP2	[invalid]	[invalid]	KP2
284	KP3	[invalid]	[invalid]	KP3
285	KP4	[invalid]	[invalid]	KP4
286	KP5	[invalid]	[invalid]	KP5
287	KP6	[invalid]	[invalid]	KP6
288	KP7	[invalid]	[invalid]	KP7
289	KP8	[invalid]	[invalid]	KP8
290	KP9	[invalid]	[invalid]	KP9
291	KP-	[invalid]	[invalid]	KP-
292	KP,	[invalid]	[invalid]	KP,
293	KP.	[invalid]	[invalid]	KP.
294	KPENTER	[invalid]	[invalid]	KPENTER
295	PGDN	[invalid]	[invalid]	PGDN
296	PGUP	[invalid]	[invalid]	PGUP
297	C-HOME	[invalid]	[invalid]	C-HOME
298	END	[invalid]	[invalid]	END
299	C-END	[invalid]	[invalid]	C-END
300	INS	[invalid]	[invalid]	INS
301	DEL	[invalid]	[invalid]	DEL
302	C-RIGHT	[invalid]	[invalid]	C-RIGHT
303	C-LEFT	[invalid]	[invalid]	C-LEFT
304	PRINT	[invalid]	[invalid]	PRINT
305	S-TAB	[invalid]	[invalid]	S-TAB
306	C-PGUP	[invalid]	[invalid]	C-PGUP
307	C-PGDN	[invalid]	[invalid]	C-PGDN
308	C-BS	[invalid]	[invalid]	C-BS
309	C-ENTER	[invalid]	[invalid]	C-ENTER
310	C-ESC	[invalid]	[invalid]	C-ESC
311	C-KP-	[invalid]	[invalid]	C-KP-
312	C-KP+	[invalid]	[invalid]	C-KP+
313	C-KP*	[invalid]	[invalid]	C-KP*
314	KP*	[invalid]	[invalid]	KP*
315	KP+	[invalid]	[invalid]	KP+
316	KP/	[invalid]	[invalid]	KP/
317	NUMLOCK	[invalid]	[invalid]	NUMLOCK
318	SCRLOCK	[invalid]	[invalid]	SCRLOCK
319	BS	[invalid]	[invalid]	BS
320	C-UP	[invalid]	[invalid]	C-UP
321	C-DOWN	[invalid]	[invalid]	C-DOWN
322	C-INS	[invalid]	[invalid]	C-INS
323	C-DEL	[invalid]	[invalid]	C-DEL
324	FIVE	[invalid]	[invalid]	FIVE
325	C-FIVE	[invalid]	[invalid]	C-FIVE
326	C-TAB	[invalid]	[invalid]	C-TAB
327	C-KP/	[invalid]	[invalid]	C-KP/
328	C-KPENTER	[invalid]	[invalid]	C-KPENTER
329	SF1	[invalid]	[invalid]	SF1
330	SF2	[invalid]	[invalid]	SF2

331	SF3	[invalid]	[invalid]	SF3
332	SF4	[invalid]	[invalid]	SF4
333	SF5	[invalid]	[invalid]	SF5
334	SF6	[invalid]	[invalid]	SF6
335	SF7	[invalid]	[invalid]	SF7
336	SF8	[invalid]	[invalid]	SF8
337	SF9	[invalid]	[invalid]	SF9
338	SF10	[invalid]	[invalid]	SF10
339	SF11	[invalid]	[invalid]	SF11
340	SF12	[invalid]	[invalid]	SF12
341	SF13	[invalid]	[invalid]	SF13
342	SF14	[invalid]	[invalid]	SF14
343	SF15	[invalid]	[invalid]	SF15
344	SF16	[invalid]	[invalid]	SF16
345	SF17	[invalid]	[invalid]	SF17
346	SF18	[invalid]	[invalid]	SF18
347	SF19	[invalid]	[invalid]	SF19
348	SF20	[invalid]	[invalid]	SF20
349	CF1	[invalid]	[invalid]	CF1
350	CF2	[invalid]	[invalid]	CF2
351	CF3	[invalid]	[invalid]	CF3
352	CF4	[invalid]	[invalid]	CF4
353	CF5	[invalid]	[invalid]	CF5
354	CF6	[invalid]	[invalid]	CF6
355	CF7	[invalid]	[invalid]	CF7
356	CF8	[invalid]	[invalid]	CF8
357	CF9	[invalid]	[invalid]	CF9
358	CF10	[invalid]	[invalid]	CF10
359	CF11	[invalid]	[invalid]	CF11
360	CF12	[invalid]	[invalid]	CF12
361	CF13	[invalid]	[invalid]	CF13
362	CF14	[invalid]	[invalid]	CF14
363	CF15	[invalid]	[invalid]	CF15
364	CF16	[invalid]	[invalid]	CF16
365	CF17	[invalid]	[invalid]	CF17
366	CF18	[invalid]	[invalid]	CF18
367	CF19	[invalid]	[invalid]	CF19
368	CF20	[invalid]	[invalid]	CF20
369	AF1	[invalid]	[invalid]	AF1
370	AF2	[invalid]	[invalid]	AF2
371	AF3	[invalid]	[invalid]	AF3
372	AF4	[invalid]	[invalid]	AF4
373	AF5	[invalid]	[invalid]	AF5
374	AF6	[invalid]	[invalid]	AF6
375	AF7	[invalid]	[invalid]	AF7
376	AF8	[invalid]	[invalid]	AF8
377	AF9	[invalid]	[invalid]	AF9
378	AF10	[invalid]	[invalid]	AF10
379	AF11	[invalid]	[invalid]	AF11
380	AF12	[invalid]	[invalid]	AF12
381	AF13	[invalid]	[invalid]	AF13
382	AF14	[invalid]	[invalid]	AF14
383	AF15	[invalid]	[invalid]	AF15
384	AF16	[invalid]	[invalid]	AF16
385	AF17	[invalid]	[invalid]	AF17
386	AF18	[invalid]	[invalid]	AF18
387	AF19	[invalid]	[invalid]	AF19
388	AF20	[invalid]	[invalid]	AF20
389	CSF1	[invalid]	[invalid]	CSF1
390	CSF2	[invalid]	[invalid]	CSF2
391	CSF3	[invalid]	[invalid]	CSF3

392	CSF4	[invalid]	[invalid]	CSF4
393	CSF5	[invalid]	[invalid]	CSF5
394	CSF6	[invalid]	[invalid]	CSF6
395	CSF7	[invalid]	[invalid]	CSF7
396	CSF8	[invalid]	[invalid]	CSF8
397	CSF9	[invalid]	[invalid]	CSF9
398	CSF10	[invalid]	[invalid]	CSF10
399	CSF11	[invalid]	[invalid]	CSF11
400	CSF12	[invalid]	[invalid]	CSF12
401	CSF13	[invalid]	[invalid]	CSF13
402	CSF14	[invalid]	[invalid]	CSF14
403	CSF15	[invalid]	[invalid]	CSF15
404	CSF16	[invalid]	[invalid]	CSF16
405	CSF17	[invalid]	[invalid]	CSF17
406	CSF18	[invalid]	[invalid]	CSF18
407	CSF19	[invalid]	[invalid]	CSF19
408	CSF20	[invalid]	[invalid]	CSF20
409	ALTA	[invalid]	[invalid]	ALTA
410	ALTB	[invalid]	[invalid]	ALTB
411	ALTC	[invalid]	[invalid]	ALTC
412	ALTD	[invalid]	[invalid]	ALTD
413	ALTE	[invalid]	[invalid]	ALTE
414	ALTF	[invalid]	[invalid]	ALTF
415	ALTG	[invalid]	[invalid]	ALTG
416	ALTH	[invalid]	[invalid]	ALTH
417	ALTI	[invalid]	[invalid]	ALTI
418	ALTJ	[invalid]	[invalid]	ALTJ
419	ALTK	[invalid]	[invalid]	ALTK
420	ALTL	[invalid]	[invalid]	ALTL
421	ALTM	[invalid]	[invalid]	ALTM
422	ALTN	[invalid]	[invalid]	ALTN
423	ALTO	[invalid]	[invalid]	ALTO
424	ALTP	[invalid]	[invalid]	ALTP
425	ALTQ	[invalid]	[invalid]	ALTQ
426	ALTR	[invalid]	[invalid]	ALTR
427	ALTS	[invalid]	[invalid]	ALTS
428	ALTT	[invalid]	[invalid]	ALTT
429	ALTU	[invalid]	[invalid]	ALTU
430	ALTV	[invalid]	[invalid]	ALTV
431	ALTW	[invalid]	[invalid]	ALTW
432	ALTX	[invalid]	[invalid]	ALTX
433	ALTY	[invalid]	[invalid]	ALTY
434	ALTZ	[invalid]	[invalid]	ALTZ
435	ALT1	[invalid]	[invalid]	ALT1
436	ALT2	[invalid]	[invalid]	ALT2
437	ALT3	[invalid]	[invalid]	ALT3
438	ALT4	[invalid]	[invalid]	ALT4
439	ALT5	[invalid]	[invalid]	ALT5
440	ALT6	[invalid]	[invalid]	ALT6
441	ALT7	[invalid]	[invalid]	ALT7
442	ALT8	[invalid]	[invalid]	ALT8
443	ALT9	[invalid]	[invalid]	ALT9
444	ALT0	[invalid]	[invalid]	ALT0
445	ALT-	[invalid]	[invalid]	ALT-
446	ALT=	[invalid]	[invalid]	ALT=

Related Topics

keyname(), keyin(), chname(), termkey()

Script Function **SYSVAR()**

Type Id Value

Purpose

To report the value of TERM system variables.

Script Usage

SETVAR *v* SYSVAR(*id*)

Parameters

id an integer assigned to a TERM internal variable. See the description for a list of assigned values.

Returns

The contents of a TERM system variable. It may return either an integer or a string, based on the value of the queried variable.

Examples

```
password encrypt("century")
? sysvar(22)
Vu<CC9Q@)%(p
password ""
? sysvar(22)
```

Description

This function is used to query the value of TERM system variables.

Following is the list of variable types id numbers and value names:

Current Configuration Settings

STR	1	config filename
STR	2	PORT
STR	3	NODE
INT	4	BAUD
STR	5	PARITY
INT	6	WORDLEN
INT	7	STOPBITS
STR	8	EMULATE
STR	9	MODE
STR	10	CHARSET
INT	11	SET WRAP
INT	12	SET BSDEL
STR	13	XPROT
STR	14	LOGIN
STR	15	LOGOUT
STR	16	STARTSERVER
STR	17	STOPSERVER
STR	18	AUTOCMD
STR	19	WRUSTR
STR	20	REMARK

STR	21	USER
STR	22	PASSWORD

Alternate Configuration Workspace Settings

STR	41	config filename
STR	42	PORT
STR	43	NODE
INT	44	BAUD
STR	45	PARITY
INT	46	WORDLEN
INT	47	STOPBITS
STR	48	EMULATE
STR	49	MODE
STR	50	CHARSET
INT	51	SET WRAP
INT	52	SET BSDEL
STR	53	XPROT
STR	54	LOGIN
STR	55	LOGOUT
STR	56	STARTSERVER
STR	57	STOPSERVER
STR	58	AUTOCMD
STR	59	WRUSTR
STR	60	REMARK
STR	61	USER
STR	62	PASSWORD

PC communications variables

INT	80	COMBASE 1 BASE
INT	81	COMBASE 1 INTR
INT	82	COMBASE 2 BASE
INT	83	COMBASE 2 INTR
INT	84	COMBASE 3 BASE
INT	85	COMBASE 3 INTR
INT	86	COMBASE 4 BASE
INT	87	COMBASE 4 INTR
INT	88	COMBASE 5 BASE
INT	89	COMBASE 5 INTR
INT	90	COMBASE 6 BASE
INT	91	COMBASE 6 INTR

File Transfer Options

INT	99	SET XFERACK
INT	100	HTIME
INT	101	STIME
INT	102	RTIME
INT	103	ITIME
INT	104	RETRIES
INT	105	RESTART
INT	106	SET XFERSTAT
INT	107	SET ESC8BIT
INT	108	SET TOUPPER
INT	109	WILDSIZE
INT	110	SET COMPRESS

INT	111	LCHAR
INT	112	LDELAY
INT	113	TDELAY
INT	114	CDELAY
INT	115	SET KERMECHO
INT	116	_KERMITEOL
INT	117	SET ALTCHK
INT	118	SET LOG
STR	119	LOGFILE
INT	120	_TXBLKSIZ
INT	121	_TXWINDOW
STR	122	FTPHOST
STR	123	FTPHOST
STR	124	FTPHOST
INT	125	SET FILECONV
INT	126	SET ADDEOF
INT	127	SET ESCCTL
STR	128	EXTGET
STR	129	EXTXFER

Terminal Options

INT	130	WRUCHAR
INT	131	PAGES
INT	132	SCROLLBACK
INT	133	SET ADDCR
INT	134	SET ADDLF
INT	135	SET MASKPAR
INT	136	SET REVDIM
INT	137	SET DIMDIM
INT	138	SET DESTBS
INT	139	SET TABEX
INT	140	SET NOSCROLL
INT	141	_TURNCHAR
INT	142	SET BLOCKMODE
INT	143	SET REVMAP
INT	144	SET EXTEND
INT	145	SET ALTKEYS
INT	146	SET SCALING
INT	147	SET SCROLLBAR
INT	148	JUMPSCROLL
STR	149	TERMID

Capture/Printer Options

STR	160	CAPTURE FILENAME with ### included
STR	161	CAPTURE FILENAME actual
STR	162	CAPTURE NONE/SPOOL/DISK/TERMINAL
INT	163	CAPTURE FLAGS bits: 0x01 append 0x02 overwrite (if not append) 0x08 flush 0x10 binary 0x20 mnemonic
STR	164	PRINTER FILENAME
STR	165	PRINTER NONE/SPOOL/DISK/TERMINAL/CAPTURE
INT	166	PRINTER FLAGS (same as capture flags)

STR	167	PRINTER SPOOLER NAME
INT	168	CAPTURE STATUS

Modem Options

STR	170	DINIT
STR	171	DPREFIX
STR	172	DSUFFIX
STR	173	DOFFHOOK
STR	174	DAUTOANS
STR	175	DANSOFF
STR	176	DESCAPE
STR	177	DHANGUP
STR	178	DOKSTR
STR	179	DRINGSTR
STR	180	DCONNECT
INT	181	DTIMOUT
INT	182	DREDIAL
INT	183	SET DDTR
STR	184	MODEM

Communications Options

INT	190	SET EXITDISC
INT	191	SET EXITDTR
STR	192	PROTOCOL
INT	193	LOCKTIME
STR	194	LOCKFILE
STR	195	NETNAME
STR	196	NETDIALOG
STR	197	NETVTNAME
INT	198	NETPORT
INT	199	NETSTIME
STR	200	DEVPREFIX
STR	201	NETWORK
INT	202	DDE ENABLE
INT	203	DDE TIMEOUT
STR	204	DDE NAME

Hardware Options

INT	210	COLOR NUMBER
STR	211	VIDOUT
STR	212	VIDCARD
STR	213	VIDUNDER
STR	214	KEYBOARD
STR	215	VIDBLINK
INT	216	XONXOFF XONCHAR
INT	217	XONXOFF XOFFCHAR
INT	218	SET EOFOPEN
INT	219	SET CONSXON
INT	220	_ESCLOOP
INT	221	VIDCARD CUSTOM AX
INT	222	VIDCARD CUSTOM BX

Color Options

STR	230	COLOR FG DEFAULT
STR	231	COLOR FG REVERSE
STR	232	COLOR FG BLINK
STR	233	COLOR FG UNDERLINE
STR	234	COLOR FG BOLD
STR	235	COLOR BG DEFAULT
STR	236	COLOR BG REVERSE
STR	237	COLOR BG BLINK
STR	238	COLOR BG UNDERLINE
STR	239	COLOR BG BOLD

General Options

STR	240	EDITOR
STR	241	FLAGS
INT	242	SET EXPLODE
INT	243	SET SOUND
STR	244	LEARN FILENAME
INT	245	SET ERRBOX
INT	246	SET VIEW
INT	247	SET CONTROL

Script Function **CHNAME()**

Purpose

To convert a number to an ascii representation.

Script Usage

SETVAR *s* CHNAME(*n*,*type*)

Parameters

n number to convert (see ranges near conversion rule).

type rule of conversion:

0 = MNEMONIC (range 0-255)
Convert to mnemonic value (e.g. hex 13=<CR>)

1 = CONTROL (range 0-255)
Convert to control char value (e.g. hex 13=^M)

2 = KEYNAME (range 0-446)
Convert to keyname value (e.g. hex 13=13)
Same as the keyname() function

Returns

The ASCII representation of the number, based on the type parameter.

Related Topics

keyid(), keyname()

Script Function **ENCRYPT()**

Purpose

To encrypt strings for use with EXMIT, ^U, and ^P commands.

Script Usage

SETVAR *s* ENCRYPT(*str*)

Parameters

str the string for encryption.

Returns

An encrypted string suitable for decrypytion with EXMIT, or the ^U or ^P DIALOG command characters.

Description

This function encrypts the passed string using TERM's encryption method. There is no decrypt() function. Decrypted data is only sent out the communications line.

Related Topics

EXMIT, LOGIN, Dialog Strings

Script Function **FCONCAT()**

Purpose

To build a full path name from path, file name and extension.

Script Usage

SETVAR *s* FCONCAT(*path*, *file*, *ext*)

Parameters

path a legal DOS path name, for example: c:.

file a legal DOS file name (8 characters or less).

ext a legal DOS file extension (3 characters or less).

Returns

A fully qualified file name.

Description

Builds a fully qualified file name from passed path, file name, and extension. If the extension is included in the file name, or not desired, use "" for ext.

TERM will accept the standard slash (/) for a path delimiter instead of the backslash (\). If you use a backslash (\), then you must put a double backslash (\\) to represent a backslash. See Strings.

Your operating system provides guidelines for valid characters and file names.

Related Topics

fext(), fhead(), Strings

Script Function **FEXT()**

Purpose

To extract the file extension from a full path name.

Script Usage

SETVAR *s* FEXT(*filename*)

Parameters

filename a full path and file name of a file.

Returns

The file extension, excluding path and file name

Description

Returns file name extension, excluding path and file name, from a full path name. The path name must be valid for DOS.

Related Topics

fext(), fhead(), fconcat()

Script Function **FHEAD()**

Purpose

To extract the file name from a full path name.

Script Usage

SETVAR *s* FHEAD(*filename*)

Parameters

filename a full path and file name of a file.

Returns

The name of the file without the extension.

Description

Returns file name, excluding extension and path, from a full path name.

Related Topics

fext(), fhead(), fconcat(), fname()

Script Function **FNAME()**

Purpose

To extract the file name+extension from a complete path name.

Script Usage

SETVAR *s* FNAME(*filename*)

Parameters

filename a full path and file name of a file.

Returns

The file name as a string.

Description

Returns file name, including extension, from a full path name. The path name must be valid for DOS.

Related Topics

fext(), fhead(), fconcat()

Script Function **GETUSERDIR()**

Purpose

To return the user's directory.

Script Usage

SETVAR *s* GETUSERDIR()

Returns

The current "user" directory, or value contained in the HOME environment variable.

Description

Returns the current "user" directory. TERM searches this directory for local .TAP files, etc. On MSDOS and Windows, it is always set to the current working directory. On all systems, if the HOME= environment variable is set, this value is returned instead.

Related Topics

CD

Script Function **KEYNAME()**

Purpose

To access the TERM key name associated with the key identification number.

Script Usage

SETVAR *s* KEYNAME(*n*)

Parameters

n an integer representing a TERM key identification number. (range 0-446)

Returns

A string representing the key name associated with n.

Description

KEYNAME() is the reverse of KEYID(). See the description under KEYID() for an explanation of key identification numbers.

Related Topics

keyid(), keyin(), chname()

Script Function **STRCONV()**

Purpose

To convert a string for display or edit in AT READ commands.

Script Usage

SETVAR *s* STRCONV(*str*, *type*)

Parameters

str a string for conversion

type one of the following:

0 binary Interpret all " and other special characters in the string (see strings), and convert these sequences to their binary equivalent.

1 ascii Convert the input string to a string containing ascii characters above space ( ) only.

2 ascii, with exceptions.
Convert the input string to a string containing ASCII characters above 20 hex and convert " to 22 hex, and ' to 27 hex.

Returns

A string filtered according to the type parameter.

Description

This function is used to prepare binary strings such as modem dialer strings to be edited in an AT READ statement, and then converted back again for TERM internal use.

Related Topics

AT, READ, Strings

Script Function **STRHEX()**

Purpose

To convert an integer to its related hexadecimal string.

Script Usage

SETVAR *s* STRHEX(*n*,*len*)

Parameters

n the number to convert

len the length of the resulting hex string desired. Values allowed are 1 through 8.

Returns

A hexadecimal string representing n. The len parameter determines its length.

NOTE: If the resulting string is longer than len, the right-most characters are returned. If the resulting hexadecimal number is shorter than len, the string is 0 filled to the left.

Description

This function converts the input number n to a hexadecimal string of length len. The leading 0x is not included as part of the output string.

Examples

For example 255 = FF hex:

```
? strhex(255,1)
F
? strhex(255,2)
FF
? strhex(255,4)
00FF
```

Script Function **FIELD()**

Purpose

To return a field from a simple delimited string.

Script Usage

SETVAR *s* FIELD(*str*, *fieldno*, *fchar*)

Parameters

str	a string, usually returned from inigetstr(), consisting of fields delimited by *fchar*.
fieldno	the number of the field desired. The item before the first delimiting character is field number 1.
fchar	the field delimiting character,which is a single character such as a comma (,).

Returns

The string value of the requested field from the string.

Description

Parses a string delimited by fchar to extract single field values.

This function is particularly useful in separating the individual values from .INI file strings such as TERM's window position, which contains the position of the upper left corner, height and width of TERM's main window in a comma-delimited string.

Examples

```
setvar x field("10,20,100,200",3,',')
? x
100
```

Related Topics

INISETSTR, inigetstr()

Script Function **INIGETSTR()**

Purpose

To read a value from a Windows .INI file.

Script Usage

SETVAR *s* INIGETSTR(*section*, *key*, *defaultstr*)

Parameters

section	the section of the .INI file to read from.
key	the key value to read.
defaultstr	the default to use if the value is undefined, or not found.

Returns

The value on the right of the equal sign from the .INI file,or the *defaultstr*.

Description

Windows .INI files contain information in the following format:

```
[section title]
key=value
```

The inigetstr() function searches for the section title and then the key. If found, it returns the value located to the right of the "=". Otherwise it returns the *defaultstr*.

This function allows a script programmer to retrieve values from .INI files.

Related Topics

INISETFILE, INISETSTR

Script Function **ISCOPY()**

Purpose

To return true if a screen area has been selected for copying to the Windows clipboard.

Script Usage

SETVAR *b* iscopy()

Returns

TRUE if a screen selection has been made for copy to the Windows clipboard.
FALSE if no selection has been made.

Description

Returns true if a screen selection is available for copy to the clipboard. Should be used before SCREEN COPY to make sure there is a selection to copy.

Related Topics

SCREEN COPY, SCREEN NOSELECT

Script Function **ISPASTE()**

Purpose

To return true if there is data in the Windows clipboard of the selected format.

Script Usage

SETVAR *b* ispaste(*format*)

Parameters

format 0 = clipboard TEXT format
1 = clipboard LINK format

Returns

TRUE if there is data of the selected format in the Windows clipboard.
FALSE if the format cannot be found in the Windows clipboard.

Description

Returns true if there is data of the selected format available to paste from the clipboard.

Related Topics

PASTE LINK, PASTE TEXT