SCRIPT LANGUAGE

Windows (WS92, version 5.4)

MS92 features a powerful script language that you can use to create scripts, which are files that contain a sequence of commands. Scripts (also called macros) are an excellent way to automate many repetitive and time consuming tasks.

For example, you can make a script that automatically dials up a computer through a modem, transmits a logon, waits for a password prompt, and submits a password. This simple script would save you time and effort in connecting to a host computer.

Scripts are contained in script files that can be run by MS92, just as other executable files are run on the PC or host.

CREATING A SCRIPT

You can create a script by automatically recording it or by manually building it:

- WS92 can *automatically record* a script by "capturing" or "storing" the keystrokes you use to perform a sequence of commands. The keystrokes are recorded and stored in a script file.
- You can create a script in WS92 by *manually building* a script file. Since script files are text files, you can create a script file with a text editor or a word processor.

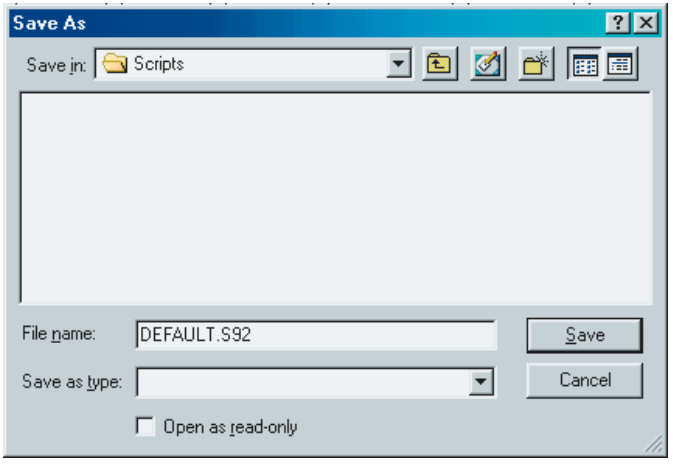
These two methods of making scripts are discussed in the next two sections.

AUTOMATICALLY RECORDING A SCRIPT

Note: This function is available only in WS92, not in DOS92.

To record a script:

1. From the File menu, select *Record Script*. The Save As dialog box appears:



2. Enter a name for the script you are about to record (make sure the extension is .S92), and click *OK*.

3. Perform the steps you would like to automate using only the keyboard. Mouse movements and clicks are *not* recorded in a script file. For example, if your script involves one of the commands on the Print menu, you must go to the menu by pressing ALT-P, not by clicking on it with the mouse.

   As you perform each function, WS92 makes a record of the keys you press in the order in which you press them. This record becomes the script file.

4. When you reach the end of recording. Press ALT-F to display the File menu. Note that there is a checkmark beside Record Script, indicating that your keystrokes are being recorded. Press T (not ALT-T) to stop recording the script.

Every keyboard action you have taken since clicking *OK* in step 2 is recorded in your new script.

See *Executing script files later* in this chapter, for directions on how to run the scripts you record.

MANUALLY BUILDING A SCRIPT

For script examples please refer to the following headings included in this chapter:

Commands

DDE Commands

Functions

WS92 Script File

MPE/iX Command File

COBOL Program

You may also view script files from our website at www.minisoft.com. From the top menu items, select *Manuals*. Under Terminal Emulation, select *Minisoft-92 Script Manual*. A PDF file will then be shown. From the pdf file, copy and paste selected script examples from the website.

The sample scripts show you how to create your own scripts in text files. If you use a word processor to write your scripts, remember to save the files in ASCII form.

Script files should always have the file extension .S92.

EXECUTING SCRIPT FILES

You can execute an MS92 script file in several ways:

- automatically, when you run MS92.
- via host commands, which are used to initiate script files.
- manually from within MS92.

In scripts executed on the PC:

- all keywords must be in UPPERCASE.
- the keyword END must be the last line of the script.

AUTOMATIC EXECUTION

Use the parameter (*scriptname)* to run a script file.

For WS92 running under Windows 3.1, append the (*scriptname)* parameter at the Command Line in the Program Item Properties dialog box in Windows Program Manager when you install MS92. Once MS92 is installed, you may change program item properties by selecting the MS92 icon (single clicking to highlight the icon rather than double clicking to run it) and then selecting Properties from the File menu of Program Manager.

For WS92 running under Windows 95, append the (*scriptname*) parameter on the Open line in the Run box.

For example, to run a script called DIALUP.S92, enter the following:

```
C:\WS92\WS92.EXE DIALUP.S92
```

If the script file itself takes parameters, add the parameters after the name of the script file. For example:

```
C:\WS92\WS92.EXE DIALUP.S92 PARM1 PARM2
```

The script file name and its parameters must come at the end of the command line. For example, you should place the name of the configuration file *before* the name of the script file:

```
C:\WS92\WS92.EXE MINISOFT.W92 DIALUP.S92
```

EXECUTION VIA HOST COMMANDS

MS92 interprets the escape sequence [ESC] &oC [cmd] [CR] as follows:

- If ***cmd*** is a script language command, MS92 will execute that command. For example:

  [ESC] & oCTELL YOU ARE NOW CONNECTED [CR]

- If ***cmd*** is not a script language command, MS92 will look for a script file with that name and run it. The PC replies S for Success, F for Failure. The PC's reply is implemented as a type 3 block transfer, meaning that it will normally require a DC1 before responding.

MANUAL EXECUTION

Select Run a script from the file menu in MS92 and enter the name of a script file or use the scroll box to select one.

SCRIPT RUNNING IN WS92

The Configure Menu Bars window is shown in the section on *Button bar and status bar configuration in WS92,* which begins in Chapter 2. The last item under Status Bar is Script Running. If you have selected this option before running a script, an **S** appears in the lower left corner of the WS92 screen whenever a script is running.

STOP SCRIPT IN WS92

The user can select the Stop Script command from the File menu in WS92 at any time the script allows input from the user.

PASSING PARAMETERS TO SCRIPTS

You can make your script files more versatile by using run-time parameters. These work in a similar way to DOS batch file parameters. For example, suppose you wrote a script file called SENDIT.S92. To have the script transmit the same file each time, write the following script:

```
LOCF MYFILE.FIL
HOSTF MYFILE
RECSIZE 256
BINARY
UPLOAD
```

To transmit a different file each time, write the following script:

```
LOCF %1
HOSTF %2
RECSIZE 256
BINARY
UPLOAD
```

You could then tell the script file to upload NEW.FIL on the PC to NEWFIL on the host by using the following command:

```
MS92 SENDIT.S92 NEW.FIL NEWFIL
```

MS92 will replace the entry %1 in the script with the first parameter (NEW.FIL), and the entry %2 with the second parameter (NEWFIL).

To write a host escape sequence to do the above, do the following:

[ESC] &oCSENDIT.S92 NEW.FIL NEWFIL [CR].

COMMAND REFERENCE

The following information contains command references that detail the proper syntax and use of all WS92 script commands and functions. Functions and commands are listed separately, in alphabetical order.

TREATMENT OF SPACES

Where parameters are delimited by parentheses, quotation marks, or separated by commas in a series, execution of the command language will ignore spaces in a command. Hence, the following are equivalent:

```
MID ("ABC", 2, 3) and  MID("ABC",2,3)
```

TYPOGRAPHICAL CONVENTIONS

Typographical conventions are used throughout this command reference to indicate proper command syntax. These conventions are as follows:

UPPERCASE

Uppercase characters indicate a keyword.

For example, in this command, HOSTF is the keyword:

```
HOSTF fname
```

lowercase italics

Characters in lowercase italics indicate a generic term for a particular item. When you issue the command, substitute the particular item for the generic term.

For example, in this command, fname is a generic term for a filename. When you issue the command, supply a particular filename, including its path if necessary:

```
HOSTF fname
```

( )

Parentheses delimit a parameter, where indicated.

For example, in this command, the string specified must be enclosed in parentheses:

```
LENGTH (string)
```

[ ]

Braces indicate that the parameter is optional.

For example, in this command, specifying off is optional:

```
LOG [OFF]
```

{}

Brackets indicate the parameter is required.

For example, in the following command, you must specify INPUT, OUTPUT, APPEND, or DELETE as a parameter (while specifying ASCII or BINARY is optional):

```
OPEN fname {INPUT | OUTPUT | APPEND | DELETE}
        AS n [ASCII | BINARY]
```

|

Vertical bar indicates a choice between two or more mutually exclusive options.

For example, in this command, if you use the ASCII | BINARY parameter, you must specify either ASCII or BINARY, not both:

```
RECEIVE LOCF FROM HOSTF [ASCII | BINARY]
```

;

Any line that starts with a semicolon is a comment line, and is ignored when the script or program that contains it is run. Comment lines are very beneficial as a way to add notes and explanations immediately adjacent to the lines of code to which the comments apply.

&

The operator for string concatenation. See the $DATE function for an example.

^

The control character, which stands for the CTRL key. The ^ control character combines with another character to form a control code.

^[

The escape code.

To display a ^ in a string as a non control character, use a double caret (^^) to neutralize it as a control character. See the following example:

```
DISPLAY "This is a ^^"
END
```

COMMANDS

ACCEPT

The ACCEPT command reads keyboard input from the user and places it in a variable. ACCEPT will read input until the user types a carriage return, unless a time limit or character other than carriage return is specified to end the command.

SYNTAX

```
ACCEPT [time] variable1 [UNTIL {string | FULL}] [LIMIT n]
[TERMINATOR variable2] [NOECHO]
```

time
Amount of time to wait for user input before canceling ACCEPT command. Format is HH:MM:SS. This parameter is optional.

variable1
The name of a variable where the user's input is to be stored. The variable will store up to 1000 characters.

UNTIL string
A character to use, instead of carriage return, to end the ACCEPT command. Specifying more than one character does not define a termination string for the command. Rather, each of the characters acts as a terminator. This parameter is optional. ACCEPT will terminate at a carriage return (^M) by default.

UNTIL FULL
Terminates the ACCEPT command when the user's input equals the value of LIMIT (below), or 1000 characters, if no LIMIT is specified. This parameter is optional.

LIMIT n
The number of characters to be read, if fewer than 1000. This parameter is optional.

TERMINATOR variable2

The name of a variable to store the character that terminates the ACCEPT command. If time is exceeded, the contents of this variable will be 0.

NOECHO

User's input is not displayed, asterisk are displayed instead (commonly used when user is entering passwords).

EXAMPLE

```
LET HEADER = "User Name"
LET PROMPT = "Please enter Your User Logon
(<user>.<acct>,<group>): "
ACCEPT USERID
LET HEADER = "Password"
LET PROMPT = "Please enter your Password: "
ACCEPT PASSWORD NOECHO
KBSPEC HP_RETRNKEY
WAITS "^Q"
TRANSMIT "hello " & USERID & "^M"
WAITS "^Q"
SEND PASSWORD
```

In the above example the first accept has a dialog box with the heading of 'User Name' and a prompt of 'Please enter Your User Logon (<user>.<acct>,<group):'. The value entered will be stored in the variable USERID.

The second accept has the heading of 'Password' and the prompt of 'Please enter your Password. The value entered will have asterisks displayed for each character because of the NOECHO parameter.

RELATED FUNCTIONS

HEADER
PROMPT

APPEND

The APPEND command is used with UPLOAD and DOWNLOAD commands to request data be appended to the end of an existing file.

SYNTAX

APPEND

EXAMPLE

```
;*****************************************
; Purge the file WS92READ from the HP3000
;*****************************************
SEND PURGE WS92READ.PUB.MINISOFT
WAITC 17
;*****************************************
; Set the ms92.msg file to uploaded
;*****************************************
LOCF C:\MINISOFT\WS92\MS92.MSG
HOSTF WS92READ.PUB.MINISOFT
ASCII
RECSIZE 90
UPLOAD
WAITC 17
;*****************************************
; Set the latest readme.txt file to be
; uploaded, this file will be appended to the
; WS92READ.PUB.MINISOFT
;*****************************************
APPEND
LOCF C:\MINISOFT\WS92\README.TXT
UPLOAD
END
```

In the above example, the file on the HP e3000 is purged and a file from the PC is uploaded creating the file with a record size of 90 bytes ASCII format. When the upload is completed a decimal 17 <DC1> trigger is sent by the HP e3000. The second upload starts and appends the second PC file to the file on the HP e3000.

Note: The RECSIZE, ASCII, or HOSTF commands do not need to be repeated for the second upload.

RELATED FUNCTIONS

ASCII
BINARY
DOWNLOAD
HOSTF
LOCF
UPLOAD
RECSIZE
RECEIVE
S
SAVINF

ASCII

The ASCII command sets the mode of the next file transfer to ASCII (or text) mode. In this mode, a carriage-return/linefeed is used as a separator between records.

SYNTAX

```
ASCII
```

EXAMPLE

```
;*****************************************
; Purge the file WS92READ from the HP3000.
;*****************************************
SEND PURGE WS92READ.PUB.MINISOFT
WAITC 17
;*****************************************
; Set the ms92.msg file to upload.
;*****************************************
LOCF C:\MINISOFT\WS92\MS92.MSG
HOSTF WS92READ.PUB.MINISOFT
ASCII
RECSIZE 90
UPLOAD
WAITC 17
;*****************************************
; Set the latest readme.txt file to be
; up loaded, this file will be appended to the
; WS92READ.PUB.MINISOFT.
;*****************************************
APPEND
LOCF C:\MINISOFT\WS92\README.TXT
UPLOAD
END
```

In the above example, the file on the HP e3000 is purged and the file from the PC is uploaded creating the file with a record size of 90 bytes ASCII format. When the upload is complete, a decimal 17 <DC1> trigger is sent by the HP e3000. The second upload starts and appends the second PC file to the file on the HP e3000.

Note: The RECSIZE, ASCII, or HOSTF commands need to be repeated for the second upload.

RELATED COMMANDS

APPEND
BINARY
DOWNLOAD
HOSTF
LOCF
UPLOAD
RECSIZE
RECEIVE
S
SAVINF

ASK

The ASK command presents a message to the user in a message field, and waits for the user to press Y for yes or N for no.

SYNTAX

ASK string

String is the text of the message. Not necessary to delimited by quotation marks.

EXAMPLE

```
CLOSE-CONNECTION
NCONNECT JAVELIN
IF $ONLINE = 0
  ASK  JAVELIN is not responding, OK to Try SUPPORT?
  IFYES TRYSUPP
ENDIF
GOTO ENDS
LABEL TRYSUPP
NCONNECT "SUPPORT"
IF $ONLINE = 0
  TELL Both Javelin and Support are not responding call MIS for
help
  EXIT
ENDIF
LABEL ENDS
END
```

In the above example, if Javelin is not responding, the ASK command will display a dialog box with Yes or No buttons. If the YES button is selected the associated command IFYES will redirect the script to the label TRYSUPP. If the No button is selected, it will go to the next script command.

RELATED COMMANDS

TELL, IFYES, LABEL, :

BACKGROUND

SYNTAX

BACKGROUND

Causes WS92's Window to minimize.

EXAMPLE

```
;************************************************
; Minimize the Window to the task bar only.
;************************************************
BACKGROUND
;************************************************
; Perform a host function such as LISTF ,2 .
;************************************************
SEND LISTF ,2
WAITC 17
;************************************************
;After the LISTF has finished restore the Window.
;************************************************
FOREGROUND
END
```

The above example will minimize the Window and perform a 'LISTF ,2'. When all files are listed, the window will be restored.

RELATED COMMANDS

FOREGROUND

BARS

The BARS command turns ON or OFF the display of the Button Bar.

SYNTAX

```
BARS {ON |OFF}
```

ON displays the Button Bar

OFF hides the Button Bar

EXAMPLE

```
;*********************************************************
; Display a dialog box to check to see if the button
; Bars should be shown. If yes go to the label TURNON,
; if not turn off the bars.
;*********************************************************
ASK Do you want to show the Button Bars?
IFYES TURNON
BARS OFF
GOTO ENDS
LABEL TURNON
BARS ON
LABEL ENDS
END
```

The above example will display a dialog box asking if you wish to show the Button Bar. Depending on your answer, it will turn on or off the Button Bars.

BAUD

The BAUD command sets the Baud of the Comm Port.

SYNTAX

BAUD {300 | 1200 | 2400 | 4800 | 9600 | 19200}

EXAMPLE

```
;*****************************************************************************
; Set the connection to Off Line.
;*****************************************************************************
OCONNECT
LABEL CONN
;*****************************************************************************
; Prompt for type of connection. Only two allowed, NSVT or Serial.
;*****************************************************************************
LET PROMPT = "Please enter the type of Connection^M(Serial, or
NSVT)
ACCEPT CONNTYPE
IF UPPER(CONNTYPE) = "NSVT"
  LET CONNT = "N"
  GOTO NSVT
ELSE
  IF UPPER(CONNTYPE) = "SERIAL"
    LET CONNT = "S"
    GOTO SERIAL
  ELSE
    TELL "Connection type must be: 'Serial', or 'NSVT'"
    GOTO CONN
  ENDIF
ENDIF
;*****************************************************************************
; If the connection is NSVT then prompt for the Node name or IP
address.
;*****************************************************************************
LABEL NSVT
LET HEADER = "ENTER HOST CONNECTION FOR NSVT"
LET PROMPT = "Please enter the HP3000 IP or Node Name"
```

```
ACCEPT NODENAME
;*****************************************************************************
; Connect using NSVT to the requested Host.
;*****************************************************************************
NCONNECT NODENAME
GOTO CONTIN
;*****************************************************************************
; If the connection is serial prompt for the comm port to use.
;*****************************************************************************
LABEL SERIAL
LET HEADER = "SERIAL/MODEM CONNECTION"
LET PROMPT = "Please Enter Comm Port Number (1-4) "
ACCEPT COMM
IF (COMM = "1") OR (COMM = "2") OR (COMM = "3") OR
(COMM = "4")
  LET COMMPORT = COMM
ELSE
  TELL "Comm port must be 1, 2, 3, or 4"
  GOTO SERIAL
ENDIF
;
LABEL BADBAUD
;*****************************************************************************
; Now that we know what comm port prompt for the Baud rate.
;*****************************************************************************
LET PROMPT = "Please Enter the Baud rate ^M 2400, 4800,
9600, or 19200"
ACCEPT BAUD
IF (BAUD = "2400") OR (BAUD = "4800") OR (BAUD = "9600")
OR (BAUD = "19200")
  LET BAUDR = BAUD
ELSE
  TELL "BAUD rate must be 2400, 4800, 9600, or 19200"
  GOTO BADBAUD
ENDIF
;*****************************************************************************
; Connect to the comm port and set the baud rate if serial.
;*****************************************************************************
CCONNECT  COMMPORT
BAUD BAUDR
LABEL CONTIN
;*****************************************************************************
; Save the new setting to the Default.w92 configuration file.
```

```
;*************************************************************************
SAVE DEFAULT.W92
END
LABEL ENDS
END
```

The above example can be used to set up the connection for the Default.W92 file. The user is prompted for the type of connection LAN via NSVT or Serial. If serial is selected then they are prompted for Comm Port to be use, and at what Baud Rate.

RELATED COMMANDS

CCONNECT

BEEP

The BEEP command sounds the PC alarm.

SYNTAX

```
BEEP
```

EXAMPLE

```
;*********************************************************
;DISPLAY The command to stop the script when a 1 is entered
;or if more than ten beeps.
;*********************************************************
DISPLAY Enter 1 to stop the beep
LET XX = 1
:RETRY
BEEP
WAIT 00:00:01 FOR "1"
IF FOUND
  GOTO ENDS
ENDIF
LET XX = XX + 1
IF XX >= 10
  GOTO ENDS
ENDIF
GOTO RETRY
:ENDS
SEND ^H
END
```

In the above example, the user's PC will make a noise (beeping sound) until a 1 is entered or after the noise has been repeated 10 times.

BINARY

The BINARY command sets the mode of the next file transfer to binary.

SYNTAX

```
BINARY
```

EXAMPLE

```
;*****************************************
; Purge the file DEFAULT from the HP3000.
;*****************************************
SEND PURGE DEFAULT.PUB.MINISOFT
WAITC 17
;*****************************************
; Set the DEFAULT.W92 configuration as the file to upload.
;*****************************************
LOCF C:\MINISOFT\WS92\DEFAULT.W92
HOSTF DEFAULT.PUB.MINISOFT
BINARY
RECSIZE 256
UPLOAD
WAITC 17
END
```

The above example will store the DEFAULT.W92 configuration file to the HP e3000 as a binary file. This can then be sent to a different PC. Contents of the file need to be in a BINARY format.

RELATED COMMANDS

ASCII	UPLOAD
APPEND	RECSIZE
DOWNLOAD	RECEIVE
HOSTF	S
LOCF	SAVINF

BLOCK_CURSOR

The BLOCK CURSOR command allows a script to change the cursor shape to either underline or block.

SYNTAX

```
BLOCK_CURSOR {ON | OFF}
```

ON for Block Cursor, *OFF* for Underline Cursor

EXAMPLE

```
LABEL CURSOR
LET HEADER = "CURSOR SETTING"
LET PROMPT = "(B) Block or (U) Underline
Cursor^MRecommend B "
ACCEPT CURS
IF UPPER(CURS) = "B"
 LET CURBLK = "ON"
 ELSE
 IF UPPER(CURS) = "U"
  LET CURBLK = "OFF"
 ELSE
  TELL "Answer must be 'B' or 'U'"
  GOTO CURSOR
 ENDIF
BLOCK_CURSOR CURBLK
ENDIF
END
```

In the above example, the user is prompted as to the cursor setting, 'B' for Block and 'U' for Underline. If anything else is entered an error message is generated and the user is asked to re-enter. Once the answer is entered correctly, the Block_Cursor command is set to *on* or *off*.

BREAK

The BREAK command sends a break signal to the host computer. This command has the same effect as pressing ALT-B within Minisoft 92.

SYNTAX

```
BREAK
```

EXAMPLE

```
;*******************************************************
; Start a application in this example it is editor
; Text in a file and then do a break followed by an
; abort. This is just an example not a recommend way to
; exit the editor.
;*******************************************************
SEND EDITOR
WAITC 17
SEND T DELTEST
WAITC 17
BREAK
WAITC 17
SEND ABORT
WAITC 17
END
```

The above example sends a break signal, waits for a host prompt, then aborts the interrupted program.

CAPS

The CAPS command allows a script to set CAP LOCK *on* or *off*.

SYNTAX

```
CAPS {ON | OFF}
```

To change all character to uppercase, set CAPS to *ON*.

To have both upper and lower case available set CAPS to *OFF*.

EXAMPLE

```
;*******************************************************
; Prompt the user asking if they want Cap Lock
; On or Off. When Cap Lock is on all Character typed as
; upper case. When off upper and lower case can be entered.
;*******************************************************
LABEL CAPLOCK
LET HEADER = "CAP LOCK SETTING"
LET PROMPT = "Cap Lock On? (Y/N)"
ACCEPT CAPL
IF UPPER(CAPL) = "Y"
  LET CAPSET = "ON"
  ELSE
  IF UPPER(CAPL) = "N"
    LET CAPSET = "OFF"
  ELSE
    TELL "Answer must be 'Y' or 'N'"
    GOTO CAPLOCK
  ENDIF
ENDIF
CAPS CAPSET
END
```

The above example will prompt and set the Cap Lock setting.

CCONNECT

The CCONNECT command sets the Comm Port.

SYNTAX

CCONNECT {1 | 2 | 3 | 4}

EXAMPLE

```
;*************************************************************************
; Set the connection to Off Line.
;*************************************************************************
OCONNECT
;*************************************************************************
; Prompt for the comm port to use.
;*************************************************************************
LABEL SERIAL
LET HEADER = "SERIAL/MODEM CONNECTION"
LET PROMPT = "Please Enter Comm Port Number (1-4) "
ACCEPT COMM
IF (COMM = "1") OR (COMM = "2") OR (COMM = "3") OR
(COMM = "4")
  LET COMMPORT = COMM
ELSE
  TELL "Comm port must be 1, 2, 3, or 4"
  GOTO SERIAL
ENDIF
;
LABEL BADBAUD
;*************************************************************************
; Now that we know what comm port prompt for the Baud rate.
;*************************************************************************
LET PROMPT = "Please Enter the Baud rate ^M 2400, 4800,
9600, or 19200"
ACCEPT BAUD
IF (BAUD = "2400") OR (BAUD = "4800") OR (BAUD = "9600")
OR (BAUD = "19200")
  LET BAUDR = BAUD
ELSE
```

```
  TELL "BAUD rate must be 2400, 4800, 9600, or 19200"
  GOTO BADBAUD
ENDIF
;*****************************************************************************
; Connect to the comm port and set the baud rate if serial.
;*****************************************************************************
CCONNECT  COMMPORT
BAUD BAUDR
;*****************************************************************************
; Save the new setting in the Default.w92 configuration file.
;*****************************************************************************
SAVE DEFAULT.W92
END
```

The above example can be used to set up the comm port *Default.W92* file. The user is prompted for the Comm Port and Baud. CCONNECT will then open the connection on that comm port/ .

RELATED COMMANDS

BAUD

CENTER

The CENTER command allows a script to center or left justify the display screen.

SYNTAX

```
CENTER {ON | OFF}
```

ON centers the display screen.

OFF left justifies the display screen

EXAMPLE

```
LABEL CENTER
LET HEADER = "CENTER DISPLAY"
LET PROMPT = "Center(C) or Left Justify (L) the ^MDisplay
within the Window?"
ACCEPT DISPLAY
IF UPPER(DISPLAY) = "C"
  LET JUSTIFY = "ON"
ELSE
  IF UPPER(DISPLAY) = "L"
    LET JUSTIFY = "OFF"
  ELSE
    TELL "Answer must be 'C' for Center or 'L' for Left Justify"
    GOTO CENTER
  ENDIF
ENDIF
CENTER JUSTIFY
END
```

The above example prompts the user, asking if the display should be centered or left justified. Takes the response as C or L and sets the display screen as requested.

CHAIN

The CHAIN command allows your script to transfer control to another script. The current script ends its execution and is removed from memory.

SYNTAX

```
CHAIN fname
```

Where *fname* is the name of a script file.

EXAMPLE

```
;CHAIN1.S92
CLOSE-CONNECTION
NCONNECT JAVELIN
IF $ONLINE = 0
  ASK  JAVELIN is not responding, OK to Try SUPPORT?
  IFYES TRYSUPP
  GOTO ENDS
ENDIF
LABEL TRYSUPP
CHAIN CHAIN2.S92
LABEL ENDS
LET VAR1 = "You are connecting to JAVELIN"
CHAIN CLOGON.S92
END
;CHAIN2.S92
NCONNECT "SUPPORT"
IF $ONLINE = 0
  TELL Both Javelin and Support are not responding call MIS for
help
  EXIT
ELSE
  LET VAR1 =  "You are connecting to SUPPORT"
  CHAIN CLOGON.S92
ENDIF
END
;CLOGIN.S92
```

```
TELL VAR1
LET HEADER = "User Name"
LET PROMPT = "Please enter Your User Log
(<user>.<acct>,<group>): "
ACCEPT USERID
LET HEADER = "Password"
LET PROMPT = "Please enter Your Password: "
ACCEPT PASSWORD NOECHO
KBSPEC HP_RETRNKEY
WAITS "^Q"
TRANSMIT "hello " & USERID & "^M"
WAITS "^Q"
SEND PASSWORD
END
```

There are three script files involved in the above example; CHAIN1.S92 CHAN2.S92 and CLOGON.S92. Normally scripts stay in memory until the WS92 is closed. With the CHAIN command, the script is remove from memory, however, variables from one script can be used in a second script.

In this example, the first script CHAIN1.S92 tries to connect to an NSVT connection using the node name JAVELIN. If JAVELIN is not responding, it will ask permission to try the node SUPPORT. If the connection is made, the third script CLOGON.S92 will use a variable set in one of the prior scripts to let you know which node you are connecting and logging on to.

RELATED COMMANDS

INVOKE
GOSUB

CHDIR

The CHDIR command allows your script to change the default directory on the PC.

SYNTAX

```
CHDIR {path name}
```

Where *path name* is the path to the new directory.

EXAMPLE

```
CHDIR C:\temp\
LOCF DELSFILE
HOSTF DELTEST
ASCII
DOWNLOAD
WAITC 17
CHDIR C:\MINISOFT\WS92\
END
```

The above example changes the path to the c:\temp\ directory for the file transfer, (the file delsfile will be sent to the C:\temp\ directory) then after the file transfer it will return to the C:\minisoft\ws92 directory.

NOTE: CD can be used in place of CHDIR.

CLOSE

The CLOSE command closes an open file or device. Files should be closed after input and output are completed, so as not to attempt to open files that may already be opened.

SYNTAX

To close a file:

CLOSE n

n Specifies the file number used to open this file. Must be in the range 1-5.

To close a device:

```
CLOSE {DISK | PRINTER}
DISK
Closes the disk currently open as the "to" device.
PRINTER
Closes the printer currently open as the "to" device.
```

EXAMPLES

```
;*************************************************************
; Close files in case they were left open from a prior process.
;*************************************************************
CLOSE 1
CLOSE 2
;*************************************************************
; Close the printer so that the log bottom can be redirected
; to disk file, followed by the open for the listfile.txt .
;*************************************************************
CLOSE PRINTER
OPEN LISTFILE.TXT
;*************************************************************
; send the HP3000 command to list the file names of the logon
; group.
;*************************************************************
send LISTF,6
```

```
;*************************************************************
; Wait for the decimal value of 10 which is a Line Feed before
; doing the log bottom, this keeps the LISTF,6 from being
; included in the list of files.
;*************************************************************
WAITC 10
LOG
WAITC 17
;*************************************************************
; Turn off the log bottom and close the disk file .
;*************************************************************
LOG OFF
CLOSE DISK
;*****************************************************************************
; Starts the second half of this script.
; Open the file from the LISTF ,6 as Input.
; Open a second file SELFILE.TXT as OUTPUT.
;*****************************************************************************
OPEN C:\MINISOFT\WS92\LISTFILE.TXT INPUT AS 1
OPEN C:\MINISOFT\WS92\SELFILE.TXT OUTPUT AS 2
;*****************************************************************************
; Read the Input file LISTFILE.TXT until a null is read indicating
End of File.
;*****************************************************************************
LABEL READAGAIN
READ 1 VAR1
IF VAR1 = ""
  GOTO EOF
ENDIF
;*****************************************************************************
; Find the period (.) prior to the group name. This will allow the
First
; and the Last Letter of the File name from the HP3000.
;*****************************************************************************
LET XX = FIND(".",VAR1) - 1
LET F = MID(VAR1,1,1)
LET L  = MID(VAR1,XX,XX)
;*****************************************************************************
; Select all files that Starts with 'A' and Ends with 'T' OR
;                    Starts with 'B' and Ends with '4' OR
;                    Starts with 'C' and Ends with '2' or '3'
; and write them to the second file SELFILE.TXT.
;*****************************************************************************
```

```
IF ((F="A")AND(L="T")) OR ((F="B")AND(L="4")) OR
((F="C")AND((L="2")OR(L="3")))
  WRITE 2 VAR1
ENDIF
GOTO READAGAIN
LABEL EOF
CLOSE 1
CLOSE 2
END
```

The above example captures the results of the LISTF,6 command to a PC file named LISTFILE.TXT. It then closes the file and reopens it as an input file, along with a new output file SELFILE.TEX.

The second part of the script reads the input file and selects all files starting with 'A' and ending with 'T', starting with 'B' and ending with '4' , or starting with 'C' and ending with either '2' or '3'. It then writes the selected files to the second file SELFIL.TXT.

RELATED COMMANDS

OPEN
READ
WRITE
LOG

CLOSE-CONNECTION

The CLOSE-CONNECTION command closes a LAN connection and changes the LAN and Serial connection to Off Line.

SYNTAX

```
CLOSE-CONNECTION
```

EXAMPLE

```
CLOSE-CONNECTION
NCONNECT JAVELIN
IF $ONLINE = 0
  ASK  JAVELIN is not responding, OK to Try SUPPORT?
  IFYES TRYSUPP
ENDIF
GOTO ENDS
LABEL TRYSUPP
NCONNECT "SUPPORT"
IF $ONLINE = 0
  TELL Both Javelin and Support are not responding call MIS for
help
  EXIT
ENDIF
LABEL ENDS
EXIT
END
```

The above example exits the WS92 terminal emulation if it can not find a host responding.

RELATED COMMANDS

QUIT
HARDEXIT

COLON

The COLON command assigns a label to a line in the script file. Other commands can cause execution of the script file to jump to this line by calling this line by its label.

SYNTAX

```
:lab
```

lab
A label for the line, up to 8 characters in length.

EXAMPLE

```
;*************************************************************
; Display a dialog box to check to see if the Button
; Bars should be shown. If yes go to the label TURNON,
; if not turn off the bars.
;*************************************************************
ASK Do you want to show the Button Bars?
IFYES TURNON
BARS OFF
GOTO ENDS
:TURNON
BARS ON
:ENDS
END
```

The above example uses the TURNON and ENDS command as labels.

RELATED COMMANDS

LABEL
GOTO
GOSUB
IFYES
IFC
ONTIMER

DEBUG

The DEBUG command runs Minisoft 92 in debug mode.

SYNTAX

DEBUG {switch}

Switch is *ON* or *OFF*.

Turns debug mode on or off.

0 = off

1 = on

Do not use this unless directed by Minisoft support.

DEL

The DEL command deletes a specified local file on the PC.

SYNTAX

```
DEL {fname}
```

fname
The name of a PC file to be deleted.

EXAMPLE

```
IF EXIST(C:\TEMP\DELTEST.TXT)
 DEL C:\TEMP\DELTEST.TXT
 TELL "C:\TEMP\DELTEST.TXT was deleted"
ELSE
 TELL "C:\TEMP\DELTEST.TXT did not exist."
ENDIF
END
```

The above example is testing for the existence of the file DELTEST.TXT in the temp directory on the C drive. If it is found, the file is deleted and a dialog box will appear with a message acknowledging its deletion. If it is not found, a dialog box will then appear displaying the message “File did not exist”.

DISCONNECT

The DISCONNECT command terminates the connection between the PC and host on a network connection. On a serial or modem connection, this command drops data transmission for two seconds.

SYNTAX

```
DISCONNECT
```

EXAMPLE

```
DISCONNECT
NCONNECT JAVELIN
IF $ONLINE = 0
  ASK  JAVELIN is not responding, OK to Try SUPPORT?
  IFYES TRYSUPP
ENDIF
GOTO ENDS
LABEL TRYSUPP
NCONNECT "SUPPORT"
IF $ONLINE = 0
  TELL Both Javelin and Support are not responding call MIS for
help
  EXIT
ENDIF
LABEL ENDS
END
```

In the above example, the command CLOSE-CONNECTION would also close the connection, except that DISCONNECT in addition to closing the LAN connection will drop a serial connection for two seconds.

RELATED COMMANDS

CLOSE-CONNECTION, TCONNECT, NCONNECT, OCONNECT, CCONNECT

DISPLAY

The DISPLAY command displays data on the terminal emulation screen. The data may be a string in quotation marks or the contents of a specified variable.

The data displays on the screen as if it were being sent from the host.

It is also the same as entering data for a block mode screen, using a tab to go from field to field and the Enter key on the number pad to send the data to host.

SYNTAX

```
DISPLAY "string" | variable
```

"string"
A string of characters, delimited by quotation marks. To include a control character in the string, precede the character with a caret (^), such as ^J for linefeed. You may use the '&' operator to concatenate (join) strings.

variable
The name of a variable.

EXAMPLE

```
DISPLAY "Password: "
ACCEPT PW NOECHO
This example prompts the user for a password.

Example 2

;*************************************************************************
: Run HP's Data entry program with a Forms file call NAMADDR
and a data
; file for the HP called Namlist. The form file has 5 fields
; Name, Address, City, State and Zip
;*************************************************************************
SEND RUN ENTRY.PUB.SYS
```

```
WAITC 17
SEND NAMADDR
WAITC 17
SEND NAMLIST
;*************************************************************************
; Open a file on the PC that has a list of name and address with
each
; field in the record separated by a semicolon.
;*************************************************************************
CLOSE 1
OPEN C:\MINISOFT\WS92\NAMADD.TXT INPUT AS 1
LABEL READAGAIN
WAITS ^[b^G^Q
READ 1 VAR1
IF VAR1 = ""
  GOTO ENDS
ENDIF
;*************************************************************************
; Fine the value of the fields, name  address city st zip in the file
; on the PC.
;*************************************************************************
LET LEN   = LENGTH(VAR1)
LET NEND  = FIND(";",VAR1)
LET NEND  = NEND - 1
LET NAME  = MID(VAR1,1,NEND)
LET NEND  = NEND + 2
LET AEND  = FIND(";",MID(VAR1,NEND,LEN))
LET AEND  = AEND + NEND - 2
LET ADDR  = MID(VAR1,NEND,AEND)
LET AEND  = AEND + 2
LET CEND  = FIND(";",MID(VAR1,AEND,LEN))
LET CEND  = CEND + AEND - 2
LET CITY  = MID(VAR1,AEND,CEND)
LET CEND  = CEND + 2
LET SEND  = FIND(";",MID(VAR1,CEND,LEN))
LET SEND  = SEND  + CEND - 2
LET STATE = MID(VAR1,CEND,SEND)
LET SEND  = SEND + 2
LET ZEND  = FIND("^M",VAR1)
IF ZEND = 0
  LET ZEND = LENGTH(VAR1)
ELSE
  LET ZEND  = ZEND - 1
```

```
ENDIF
LET ZIP   = MID(VAR1,SEND,ZEND)
;*************************************************************************
; After finding the fields the display will put them into the data file
; on the HP. If the field is not full a Tab moves you to the next
field.
; A enter key will terminate the entry of that record.
;*************************************************************************
DISPLAY NAME
IF LENGTH(NAME) < 21
  KBSPEC HP_TABKEY
ENDIF
DISPLAY ADDR
IF LENGTH(ADDR) < 20
  KBSPEC HP_TABKEY
ENDIF
DISPLAY CITY
IF LENGTH(CITY) < 15
  KBSPEC HP_TABKEY
ENDIF
DISPLAY STATE
IF LENGTH(STATE) < 2
  KBSPEC HP_TABKEY
ENDIF
DISPLAY ZIP
KBSPEC HP_ENTERKEY
GOTO READAGAIN
LABEL ENDS
;*************************************************************************
; Function Key 7 will put you into Browse mode.
; Function Key 1 will locate the first record in the input.
;*************************************************************************
KBSPEC HP_F7KEY
WAITS ^[b^Q
KBSPEC HP_F1KEY
END
```

The above script file is an example of using display in block mode.

RELATED COMMANDS

KBSPEC

DOWNLOAD

The DOWNLOAD command transfers a file from the host to the PC.

SYNTAX

```
DOWNLOAD
```

EXAMPLE

```
 LOCF C:\WINWORD\README.TXT
HOSTF MS92305.README.MINISOFT
ASCII
DOWNLOAD
```

The above example transfers the host file MS92305.README.MINISOFT to the PC, where it will be called README.TXT in the WINWORD directory on the C drive.

RELATED COMMANDS

To use the DOWNLOAD command, you must define LOCF and HOSTF in the script file. Also specify ASCII or BINARY before issuing the DOWNLOAD command.

To name LOCF, HOSTF, and file transfer method as parameters of the download operation, use the RECEIVE command.

Use the APPEND command with the DOWNLOAD command to append data to the end of an existing file. For example:

```
LOCF C:\WINWORD\README.TXT
HOSTF MS92305.README.MINISOFT
APPEND
ASCII
DOWNLOAD
```

ELSE

The ELSE command marks the start of an execution if the previous IF condition is not true.

SYNTAX

ELSE
command.

command
Any valid script command.

EXAMPLE

```
LABEL CURSOR
LET HEADER = "CURSOR SETTING"
LET PROMPT = "(B) Block or (U) Underline
Cursor^MRecommend B "
ACCEPT CURS
IF UPPER(CURS) = "B"
  LET CURBLK = "ON"
  ELSE
  IF UPPER(CURS) = "U"
    LET CURBLK = "OFF"
  ELSE
    TELL "Answer must be 'B' or 'U'"
    GOTO CURSOR
  ENDIF
BLOCK_CURSOR CURBLK
ENDIF
  END
```

RELATED COMMANDS

IF, ENDIF

END

The END command marks the end of a script file. Script files must end with this command.

SYNTAX

```
END
```

EXAMPLE

```
LABEL CURSOR
LET HEADER = "CURSOR SETTING"
LET PROMPT = "(B) Block or (U) Underline
Cursor^MRecommend B "
ACCEPT CURS
IF UPPER(CURS) = "B"
  LET CURBLK = "ON"
  ELSE
  IF UPPER(CURS) = "U"
    LET CURBLK = "OFF"
  ELSE
    TELL "Answer must be 'B' or 'U'"
    GOTO CURSOR
  ENDIF
BLOCK_CURSOR CURBLK
ENDIF
 END
```

ENDIF

The ENDIF command marks the end of an IF statement.

SYNTAX

```
ENDIF
```

EXAMPLE

```
LABEL CURSOR
LET HEADER = "CURSOR SETTING"
LET PROMPT = "(B) Block or (U) Underline
Cursor^MRecommend B "
ACCEPT CURS
IF UPPER(CURS) = "B"
 LET CURBLK = "ON"
 ELSE
 IF UPPER(CURS) = "U"
  LET CURBLK = "OFF"
 ELSE
  TELL "Answer must be 'B' or 'U'"
  GOTO CURSOR
 ENDIF
BLOCK_CURSOR CURBLK
ENDIF
  END
```

RELATED COMMANDS

IF
ELSE

ERASE

The ERASE command deletes a specified local file on the PC.

SYNTAX

```
ERASE {fname}
```

fname
The name of a PC file to be deleted.

EXAMPLE

```
IF EXIST(C:\TEMP\DELTEST.TXT)
 ERASE C:\TEMP\DELTEST.TXT
 TELL "C:\TEMP\DELTEST.TXT was deleted"
ELSE
 TELL "C:\TEMP\DELTEST.TXT did not exist."
ENDIF
END
```

The above example is testing for the existence of the file DELTEST.TXT in the Temp directory on the C drive. If it is found, the file is deleted and a dialog box will appear with a message acknowledging its deletion. If it is not found, a dialog box will then appear displaying the message “File did not exist”.

RELATED COMMANDS

DEL

EXIT

The EXIT command exits Minisoft 92 and sets the DOS error level if communicating over a Serial port. The user will remain logged on to the host, but if they are a Network user they are disconnected.

SYNTAX

```
EXIT
```

EXAMPLE

```
CLOSE-CONNECTION
NCONNECT JAVELIN
IF $ONLINE = 0
  ASK  JAVELIN is not responding, OK to Try SUPPORT?
  IFYES TRYSUPP
ENDIF
GOTO ENDS
LABEL TRYSUPP
NCONNECT "SUPPORT"
IF $ONLINE = 0
  TELL Both Javelin and Support are not responding call MIS for
help
  EXIT
ENDIF
LABEL ENDS
EXIT
END
```

The above example exits WS92 terminal emulation if it can not find a host that is responding.

RELATED COMMANDS

```
QUIT
HARDEXIT
```

FOREGROUND

The FOREGROUND command causes WS92 to come into 'focus'. A FOREGROUND script or host command can be used to get the users attention by changing the status of a window from background to foreground.

EXAMPLE

```
;*************************************************
; Minimize the Window to the task bar only.
;*************************************************
BACKGROUND
;*************************************************
; Perform a host function such as LISTF ,2 .
;*************************************************
SEND LISTF ,2
WAITC 17
;*************************************************
;After the LISTF has finished, restore the Window.
;*************************************************
FOREGROUND
END
```

The above example minimizes the Window and performs a 'LISTF ,2' . When all files are listed the window will then be restored.

RELATED COMMANDS

BACKGROUND

GOSUB

The GOSUB command executes a subroutine that begins on the line following the specified LABEL command.

SYNTAX

GOSUB LABEL

LABEL
Labels the starting line of the subroutine.

EXAMPLE

```
;*************************************************************************
;    This script show that a variable can be asked for, checked,
;    and then imbedded as part of a file name.
;    The GOSUB and file transfer is used.
;*************************************************************************
LABEL START
;*************************************************************************
;    Clean up the screen with a Home and Clear display.
;*************************************************************************
KBSPEC HP_HOMEUKEY
KBSPEC HP_CLRDKEY
;*************************************************************************
;    Calls a Subroutine that will Prompt for a four digit
date(mmdd).
;*************************************************************************
GOSUB GETDATE
IF DATEOK <> "OK"
  GOTO START
ENDIF
;*************************************************************************
;    Calls a Subroutine that will build a file using the date obtain.
;*************************************************************************
GOSUB BLDFILEN
IF FILEOK <> "OK"
  GOTO START
ENDIF
```

```
;*************************************************************************
;    Calls a Subroutine that will check to see if the file is
;    on the PC.
;*************************************************************************
GOSUB CHECKFILE
IF FILEOK <> "OK"
  GOTO START
ENDIF
;*************************************************************************
;    Calls a Subroutine that does the file transfer to the Host.
;*************************************************************************
GOSUB XFER
;*************************************************************************
;    Check to see if you want to transfer a different date file.
;*************************************************************************
ASK DO YOU WANT DO AN OTHER FILE?
IFYES START
GOTO ENDS
;*************************************************************************
;  The subroutine that prompt for a four digit date (mmyy).
;*************************************************************************
LABEL GETDATE
LET HEADER="File Date"
LET PROMPT="Enter the date that needs to be embedded in
^Mthe local file name"
ACCEPT FILEDATE LIMIT 4
LET DATELEN=LENGTH(FILEDATE)
IF DATELEN <> 4
   TELL "Date must be four characters long. "
   GOSUB ASKQUIT
   LET DATEOK = "BAD"
ELSE
   LET DATEOK = "OK"
ENDIF
RETURN
;*************************************************************************
;    The subroutine that builds the PC file name.
;*************************************************************************
LABEL BLDFILEN
LET LOCFILE  = "C:\TEMP\TEST" & FILEDATE  & ".TXT"
DISPLAY LOCFILE
ASK IS THIS THE CORRECT FILE?
IFYES CONT
```

```
LET FILEOK = "BAD"
GOSUB ASKQUIT
RETURN
LABEL CONT
LET FILEOK = "OK"
RETURN
;*************************************************************************
;     Subroutine to check if the file is on the PC.
;*************************************************************************
LABEL CHECKFILE
IF EXIST(LOCFILE)
  LET FILEOK = "OK"
  RETURN
ELSE
  LET FILEOK = "BAD"
  TELL "PC file not found"
  GOSUB ASKQUIT
  RETURN
ENDIF
;*************************************************************************
;     Subroutine Prompts to see if you want to quit or not.
;*************************************************************************
LABEL ASKQUIT
ASK  DO YOU WISH TO QUIT?
IFYES ENDS
RETURN
;*************************************************************************
;     Subroutine that Starts the file transfer.
;*************************************************************************
LABEL XFER
DISPLAY "^M^JThank you the file transfer will begin."
LOCF LOCFILE
HOSTF "TEST" & FILEDATE & ".DEL.MINISOFT"
ASCII
RECSIZE 80
UPLOAD
RETURN
;*************************************************************************
; This is the Label that the File Transfer competed and you do
not
; addition files to transfer. It also is the label that use to quit
; when an error has occurred and you want to Quit.
;*************************************************************************
```

```
LABEL ENDS
END
```

RELATED COMMANDS

RETURN
Colon (:)
LABEL

GOTO

The GOTO command executes a jump to a specified line in the script file.

SYNTAX

```
GOTO lab
```

lab
Label for the line.

EXAMPLE

```
;*************************************************************
;DISPLAY The command to stop the script when a 1 is entered
;or if more than ten beeps.
;*************************************************************
DISPLAY Enter 1 to stop the beep
LET XX = 1
:RETRY
BEEP
WAIT 00:00:01 FOR "1"
IF FOUND
  GOTO ENDS
ENDIF
LET XX = XX + 1
IF XX >= 10
  GOTO ENDS
ENDIF
GOTO RETRY
:ENDS
SEND ^H
END
```

In the above example, the user's PC will make a noise (beeping sound) until a 1 is entered or the noise is repeated up to 10 times.

RELATED COMMANDS

Colon (:), LABEL

HARDEXIT

The HARDEXIT command exits Minisoft 92 and sets the DOS error level if communicating over a Serial port. The user will remain logged on to the host, if they are a Network user they are disconnected.

SYNTAX

```
HARDEXIT [n]
```

n

DOS error level. This parameter is optional (Default is 0).

EXAMPLE

```
CLOSE-CONNECTION
NCONNECT JAVELIN
IF $ONLINE = 0
  ASK  JAVELIN is not responding, OK to Try SUPPORT?
  IFYES TRYSUPP
ENDIF
GOTO ENDS
LABEL TRYSUPP
NCONNECT "SUPPORT"
IF $ONLINE = 0
  TELL Both Javelin and Support are not responding call MIS for
help
  HARDEXIT
ENDIF
LABEL ENDS
HARDEXIT
END
```

The above example logs the user off the host and exits Minisoft 92.

RELATED COMMANDS

EXIT, QUIT

HEADER

SYNTAX

LET HEADER = string value

EXAMPLE

```
LET HEADER = "User Name"
LET PROMPT = "Please enter Your User Logon
(<user>.<acct>,<group>): "
ACCEPT USERID
LET HEADER = "Password"
LET PROMPT = "Please enter your Password: "
ACCEPT PASSWORD NOECHO
KBSPEC HP_RETRNKEY
WAITS "^Q"
TRANSMIT "hello " & USERID & "^M"
WAITS "^Q"
SEND PASSWORD
```

The first header sets a dialog box with the heading 'User Name' and a prompt of “Please enter Your User Logon (<user>.<acct>,<group):”. The value entered will be stored in the variable USERID.

The second header sets a dialog box with the heading 'Password' and a prompt of “Please enter your Password”. The value entered will have asterisks displayed for each character because of the NOECHO parameter.

RELATED FUNCTIONS

ACCEPT

HOSTF

The HOSTF command names a file on the host for file transfer.

When downloading, this is the file being transferred to the PC.

When uploading, this is what the PC file will be called on the host.

SYNTAX

```
HOSTF fname [TEMP]
```

fname
The name of the host file.

[TEMP]
Optional parameter that identifies the host file as a temporary file.

EXAMPLE

```
;*****************************************
; Purge the file WS92READ from the HP e3000.
;*****************************************
SEND PURGE WS92READ.PUB.MINISOFT
WAITC 17
;*****************************************
; Set the ms92.msg file to upload.
;*****************************************
LOCF C:\MINISOFT\WS92\MS92.MSG
;*****************************************
; Set the Host file name.
;*****************************************
HOSTF WS92READ.PUB.MINISOFT
ASCII
RECSIZE 90
UPLOAD
WAITC 17
;*****************************************
; Set the latest readme.txt file to be
; up loaded, this file will be appended to the
; WS92READ.PUB.MINISOFT .
```

```
;*****************************************
APPEND
LOCF C:\MINISOFT\WS92\README.TXT
UPLOAD
END
```

In the above example, the file on the HP e3000 is purged and the file from the PC is uploaded, creating a file with the record size of 90 bytes ASCII format. When the upload is complete, a decimal 17 <DC1> trigger is sent by the HP e3000. The second upload can start and will append the second PC file to the file on the HP e3000.

Note: The RECSIZE, ASCII, or HOSTF commands DO NOT need to be repeated for the second upload.

RELATED COMMANDS

APPEND
ASCII
BINARY
DOWNLOAD
LOCF
UPLOAD
RECSIZE
RECEIVE
S
SAVINF

HOSTPORT

The HOSTPORT command allows a script to set a TELNET port to a specified value.

SYNTAX

HOSTPORT{n}

Sets the Telnet port number to a value of n.

EXAMPLE

```
;*************************************************************
; Load the DEFAULT.W92 file from the c:\minisoft\ws92\folder.
;*************************************************************
LOAD c:\minisoft\ws92\DEFAULT.W92
;*************************************************************
; Set the TCPIP port to 23
; Set a variable to a String value of "192.10.10.10" the
; IP address of a host that you want to do a Telnet connection .
;*************************************************************
HOSTPORT 23
LET NODEIP = "192.10.10.10"
;*************************************************************
; Set the telnet connection to the new Telnet node.
;*************************************************************
TCONNECT NODEIP
;*************************************************************
; Save the configuration to a new configuration file call
; TELNET.W92 .
; Load the new configuration file .
;*************************************************************
SAVE c:\minisoft\ws92\TELNET.W92
LOAD c:\minisoft\ws92\TELNET.W92
END
```

RELATED COMMANDS

TCONNECT

HOSTPRINT

The HOSTPRINT command prints the specified local file to a printer attached to the host. This command is the Print File on Host Printer command from the File menu in Minisoft 92 for Windows.

The printer where HOSTPRINT sends the file is the 'Host Printer Name' configured in Minisoft 92 as part of the Host File Transfer Configuration.

SYNTAX

```
HOSTPRINT fname
```

fname
The name of the local file to print. This file must reside on the PC.

EXAMPLE

```
;*************************************************************************
;    This asks for a PC file name. Checks for the File and if
found
;    transfers the file to the Printer configured in the File
;    Transfer setup menu as the 'Host Printer Name' .
;*************************************************************************
LABEL START
;*************************************************************************
;    Clean up the screen with a Home and Clear display
;*************************************************************************
KBSPEC HP_HOMEUKEY
KBSPEC HP_CLRDKEY
;*************************************************************************
;    Calls a Subroutine that will Prompt for a PC File name.
;*************************************************************************
GOSUB GetFileN
;*************************************************************************
;    Calls a Subroutine that will check to see if the file is
;    on the PC.
;*************************************************************************
GOSUB CHECKFILE
```

```
 IF FILEOK <> "OK"
   GOTO START
 ENDIF
;*************************************************************************
;     Calls a Subroutine that send the file to the HP Host printer.
;*************************************************************************
 GOSUB PRINT
;*************************************************************************
;     Check to see if you want to different PC file.
;*************************************************************************
 ASK DO YOU WANT DO AN OTHER FILE?
 IFYES START
 GOTO ENDS
;*************************************************************************
;   The subroutine that prompt for a PC File Name to be printed.
;*************************************************************************
 LABEL GETFILEN
 LET HEADER="PC File Name"
 LET PROMPT="Enter the Full Path of the PC file^M to be
Printed on the HP"
 ACCEPT FILENAME
 RETURN
;*************************************************************************
;     Subroutine to check if the file is on the PC.
;*************************************************************************
 LABEL CHECKFILE
 IF EXIST(FILENAME)
   LET FILEOK = "OK"
   RETURN
 ELSE
   LET FILEOK = "BAD"
   TELL "PC file not found"
   GOSUB ASKQUIT
   RETURN
 ENDIF
;*************************************************************************
;      Subroutine prompts to see if you want to quit or not.
;*************************************************************************
 LABEL ASKQUIT
 ASK  DO YOU WISH TO QUIT?
 IFYES ENDS
 RETURN
;*************************************************************************
```

```
;     Subroutine that Starts the transfer of the PC file to
;     the HP Printer.
;*************************************************************************
LABEL PRINT
DISPLAY "^M^JThank you the file will be sent to the printer"
HOSTPRINT FILENAME
WAITC 17
XMITC 13
WAITC 17
RETURN
;*************************************************************************
; This is the Label that the File Transfer to when you do not
want
; addition PC files to be printed. It also is the label that use to
quit
; when an error has occurs and you reply yes to Quit.
;*************************************************************************
LABEL ENDS
END
```

IF

The IF command Tests for a condition and executes commands if the condition is true. If the condition is not true, execution jumps to the next ELSE or ENDIF command.

Conditions are stated with logical expressions, such as equal to (=), not equal to (<>), greater than (>), less than (<), less than or equal to (<=), and greater than or equal to (>=). Expressions may be linked with the logical operators AND and OR. For multiple comparisons between strings, numbers, and/or variables, the comparisons must be enclosed in parentheses.

SYNTAX

```
IF condition
   command
```

condition
An expression, comparison, or logical operation.

command
The command executed if the condition is true.

EXAMPLE

```
;*****************************************************************************
;
; Set the connection to Off Line.
;*****************************************************************************
;
OCONNECT
;*****************************************************************************
;
; Set the type of connection only to Serial.
;*****************************************************************************
;
 LABEL SERIAL
LET HEADER = "SERIAL/MODEM CONNECTION"
LET PROMPT = "Please Enter Comm Port Number (1-4) "
ACCEPT COMM
IF (COMM = "1") OR (COMM = "2") OR (COMM = "3") OR
(COMM = "4")
  LET COMMPORT = COMM
```

```
ELSE
  TELL "Comm port must be 1, 2, 3, or 4"
  GOTO SERIAL
ENDIF
;
LABEL BADBAUD
;*************************************************************************
; Now that we know what comm port prompt for the Baud rate.
;*************************************************************************
LET PROMPT = "Please Enter the Baud rate ^M 2400, 4800,
9600, or 19200"
ACCEPT BAUD
IF (BAUD = "2400") OR (BAUD = "4800") OR (BAUD = "9600")
OR (BAUD = "19200")
  LET BAUDR = BAUD
ELSE
  TELL "BAUD rate must be 2400, 4800, 9600, or 19200"
  GOTO BADBAUD
ENDIF
;*************************************************************************
; Connect to the comm port and set the Baud Rate if Serial.
;*************************************************************************
CCONNECT  COMMPORT
BAUD BAUDR
;*************************************************************************
; Save the new setting in the Default.w92 configuration file.
;*************************************************************************
SAVE DEFAULT.W92
END
```

The above example can be used to setup the connection for the *Default.W92* file. The user is prompted for the Comm Port to be used and at what Baud Rate.

RELATED COMMANDS

ELSE
ENDIF
IFYES

IFYES

The IFYES command causes execution to jump to a line with the specified label if the user answers Y to the last ASK command.

EXAMPLE

```
CLOSE-CONNECTION
NCONNECT JAVELIN
IF $ONLINE = 0
  ASK  JAVELIN is not responding, OK to Try SUPPORT?
  IFYES TRYSUPP
ENDIF
GOTO ENDS
LABEL TRYSUPP
NCONNECT "SUPPORT"
IF $ONLINE = 0
  TELL Both Javelin and Support are not responding call MIS for
help
  EXIT
ENDIF
LABEL ENDS
END
```

IFYES is always preceded by an ASK command and has a LABEL name to go to.

RELATED COMMAND

ASK
LABEL

INVOKE

The INVOKE command allows your script to temporarily transfer control to another script. The current script suspends its execution and resumes execution at the next line when the called script ends. Do not use the RETURN command to exit from an invoked script.

SYNTAX

```
INVOKE fname
```

Where *fname* is the name of a script file.

EXAMPLE

Main Script

```
;*************************************************************************
;    This script show that a variable can be asked for, checked,
;    and then imbedded as part of a file name.
;    The INVOKE is used to call different Scripts.
;*************************************************************************
LABEL START
;*************************************************************************
;    Clean up the screen with a Home and Clear display.
;*************************************************************************
KBSPEC HP_HOMEUKEY
KBSPEC HP_CLRDKEY
;*************************************************************************
;    Calls a Getdate Script that will Prompt for a four digit
date(mmdd) .
;*************************************************************************
INVOKE GETDATE.S92
IF DATEOK <> "OK"
  GOSUB ASKQUIT
  GOTO START
ENDIF
;*************************************************************************
;    Calls a BLDFILEN Script that will build a file using the date
obtain.
```

```
;*************************************************************************
INVOKE BLDFILEN.S92
IF FILEOK <> "OK"
  GOSUB ASKQUIT
  GOTO START
ENDIF
;*************************************************************************
;    Calls a CHECKFILE script that will check to see if the file is
;    on the PC.
;*************************************************************************
INVOKE CHECKFILE.S92
IF FILEOK <> "OK"
  GOSUB ASKQUIT
  GOTO START
ENDIF
;*************************************************************************
;    Calls XFER script  that does the file transfer to the Host.
;*************************************************************************
INVOKE XFER.S92
;*************************************************************************
;    Check to see if you want to transfer a different date file.
;*************************************************************************
ASK DO YOU WANT DO AN OTHER FILE?
IFYES START
GOTO ENDS
;*************************************************************************
;     Subroutine Prompts to see if you want to quit or not.
;*************************************************************************
LABEL ASKQUIT
ASK  DO YOU WISH TO QUIT?
IFYES ENDS
RETURN
LABEL ENDS
END
```

Second Script GETDATE.S92

```
;*************************************************************************
;  A Script that prompt for a four digit date (mmyy).
;*************************************************************************
LABEL GETDATE
LET HEADER="File Date"
LET PROMPT="Enter the date that needs to be embedded in
^Mthe local file name"
```

```
ACCEPT FILEDATE LIMIT 4
LET DATELEN=LENGTH(FILEDATE)
IF DATELEN <> 4
  TELL "Date must be four characters long (MMYY). "
  LET DATEOK = "BAD"
ELSE
  LET DATEOK = "OK"
ENDIF
END
```

Third Script BLDFILEN.S92

```
;*************************************************************************
;    The SCRIPT  that builds the PC file name.
;*************************************************************************
LABEL BLDFILEN
LET LOCFILE  = "C:\TEMP\TEST" & FILEDATE  & ".TXT"
DISPLAY LOCFILE
ASK IS THIS THE CORRECT FILE?
IFYES CONT
LET FILEOK = "BAD"
GOTO ENDS
LABEL CONT
LET FILEOK = "OK"
:ENDS
END
```

Fourth Script CHECKFILE.S92

```
;*************************************************************************
;     Script to check if the file is on the PC.
;*************************************************************************
LABEL CHECKFILE
IF EXIST(LOCFILE)
  LET FILEOK = "OK"
ELSE
  LET FILEOK = "BAD"
  TELL "PC file not found"
ENDIF
END
```

RELATED COMMANDS

GOSUB, CHAN

KBNORM

The KBNORM command emulates a single keyboard character; acts as though a key had been pressed on the keyboard. Use the ASCII decimal value of the character.

SYNTAX

```
KBNORM c
```

c

An ASCII character decimal value.

EXAMPLE

```
KBNORM 83
KBNORM 72
KBNORM 79
KBNORM 87
KBNORM 77
KBNORM 69
KBNORM 13
END
```

The above script send a SHOWME plus a carriage return to the host.

RELATED COMMANDS

XMITC
KBSPEC
KBSTRING

KBSPEC

The KBSPEC command emulates a single keyboard character; acts as though a key had been pressed on the keyboard. Use the ASCII decimal value of the character.

SYNTAX

KBSPEC {HP or VT Key Constants}]

EXAMPLE

```
XMITC 13
KBNORM 83
KBNORM 72
KBNORM 79
KBNORM 87
KBNORM 77
KBNORM 69
KBSPEC HP_RETRNKEY
END
```

The above script sends a SHOWME plus a carriage return to the Host.

RELATED COMMANDS

XMITC, KBNORM, KBSTRING

HP Key Constant

HP_BRAKEKEY	HP_BSKEY
HP_BTABKEY	HP_CLRDKEY
HP_COMMAKEY	HP_CURSDKEY
HP_CURSLKEY	HP_CURSRKEY
HP_CURSUKEY	HP_DELCKEY
HP_DELKEY	HP_DELLKEY
HP_ENTERKEY	HP_F1KEY
HP_F2KEY	HP_F3KEY
HP_F4KEY	HP_F5KEY
HP_F6KEY	HP_F7KEY

HP_F8KEY	HP_HOMEDKEY
HP_HOMEUKEY	HP_INSCKEY
HP_INSLKEY	HP_INSWRAPKRY
HP_MENUKEY	HP_NEXTKEY
HP_PREVKEY	HP_PRINTKEY
HP_RETRNKEY	HP_ROLLDKEY
HP_ROLLUKEY	HP_SRSETKEY
HP_STOPKEY	HP_TABKEY
HP_UMENUKEY	HP_USERKEY

VT Key Constant

VT_BSKEY	VT_BTABKEY
VT_COMMA_KEY	VT_CURSDKEY
VT_CURSLKEY	VT_CURSRKEY
VT_CURSUKEY	VT_DELKEY
VT_F1KEY	VT_F2KEY
VT_F3KEY	VT_F4KEY
VT_F5KEY	VT_F6KEY
VT_F7KEY	VT_F8KEY
VT_F9KEY	VT_F10KEY
VT_F11KEY	VT_F12KEY
VT_F13KEY	VT_F14KEY
VT_F15KEY	VT_F16KEY
VT_F17KEY	VT_F18KEY
VT_F19KEY	VT_F20KEY
VT_FINKKEY	VT_INSERTKEY
VT_NEXTKEY	VT_NUM_0_KEY
VT_NUM_1_KEY	VT_NUM_2_KEY
VT_NUM_3_KEY	VT_NUM_4_KEY
VT_NUM_5_KEY	VT_NUM_6_KEY
VT_NUM_7_KEY	VT_NUM_8_KEY
VT_NUM_9_KEY	VT_PF1KEY
VT_PF2KEY	VT_PF3KEY
VT_PF4KEY	VT_PREVKEY
VT_REMOVEKEY	VT_RETRNKEY
VT_SELECTKEY	VT_SHIFTF6KEY
VT_SHIFTF7KEY	VT_SHIFTF8KEY
VT_SHIFTF9KEY	VT_SHIFTF10KEY
VT_SHIFTF11KEY	VT_SHIFTF12KEY
VT_SHIFTF13KEY	VT_SHIFTF14KEY
VT_SHIFTF15KEY	VT_SHIFTF16KEY
VT_SHIFTF17KEY	VT_SHIFTF18KEY
VT_SHIFTF19KEY	VT_SHIFTF20KEY
VT_TABKEY	

KBSTRING

The KBSTRING command emulates a string as though it had been typed on the keyboard.

SYNTAX

```
KBSTRING string
```

string
Any valid keyboard string.

EXAMPLE

```
KBSTRING HELLO MGR.MINISOFT
END
```

RELATED COMMANDS

XMITS

KEYMAP BACKSPACE TO

The KEYMAP BACKSPACE TO command maps function key *n* to the string value in quotes. A caret (^) preceding a character changes the character to a Control + the character.

SYNTAX

KEYMAP BACKSPACE TO <string>

KEYMAP FN TO

The KEYMAP FN TO command maps function key *n* to the string value in quotes. A caret (^) preceding a character changes the character to a Control + the character.

SYNTAX

KEYMAP Fn TO <string>

LABEL

The LABEL command assigns a label to a line in the script file. Other commands can cause execution of the script file to jump to this line by calling it by its label.

SYNTAX

```
LABEL lab
```

lab
A label for the line. Up to 9 characters in length.

EXAMPLE

```
LABEL DIALMODEM
```

In the above example, the line has been labeled DIALMODEM.

RELATED COMMANDS

The LABEL command works exactly like the colon (:).

To jump to a labeled line, use the GOTO or GOSUB commands.

To jump to a labeled line under certain conditions use the IF, IFYES, or ONTIMER commands.

LENGTH

The LENGTH command returns the number of characters in a specified string.

SYNTAX

```
LENGTH (string)
```

string

Maybe a variable or a string delimited by quotation marks. To include a control character in the string, precede the character with a caret (^), such as ^J for linefeed. You may use the '&' operator to concatenate (join) strings.

EXAMPLE

```
LET Var1 = "Last Record"
LET Var2 = LENGTH(Var1)
LET Var3 = "The length of Var1 is: "
LET Var3 = Var3 & STRING(Var2)
TELL Var3
END
```

In the above example, a variable named Var2 is defined as the length of the string, "Last Record" which is 11. Var3 sent the string "The length of Var1 is: " and using the STRING function, the integer value of Var2 was concatenated to the string of Var3. The TELL command will then display a dialog box with the message "The length of Var1 is: 11".

RELATED FUNCTIONS

FIND
STRING
TELL

LET

The LET command stores the results of an expression in a variable. It also provides a way of mapping a string to the PC's special keys (such as cursor keys, home key, etc.).

SYNTAX

```
LET variable = expression
```

variable

The name of the variable where the result is stored (Up to 8 characters). Any characters are valid, except for reserved words (other script command names). There are predefined variables for certain PC keys; see the following list below the heading *Predefined Variables for PC Keys*.

expression

Numeric or string expression. Numeric expressions can contain numeric constants and functions combined with the following operators: multiply (*), divide (/), add (+). and subtract (-). String expressions can contain string constants or calls to string functions. You may also use the '&' operator to concatenate (join) strings. The variable will store up to 80 characters.

EXAMPLE

```
LET Var1 = "HELLO MGR.MINISOFT^M"
TRANSMIT Var1
```

In the above example, a variable named Var1 is defined as a logon, which may be sent to the host with the TRANSMIT command.

```
LET Var1 = LENGTH ("Last Record")
LET Var2 = Var1 + 2
```

In this example, a variable named Var1 is the length of the string, "Last Record" while a variable named Var2 is the sum of the value of Var1 and 2, or 13.

```
LET CURSLKEY="This is the left key."
```

In this example pressing the left arrow cursor key transmits the string, "This is the left key." For a list of predefined PC key variables, see the list below.

PREDEFINED VARIABLES FOR PC KEYS

The following variables are predefined for the PC keys and key combinations shown:

PC Key	Predefined Variable
ENTER	"ENTERKEY"
RETURN	"RETRNKEY"
TAB	"TABKEY"
SHIFT-TAB	"BTABKEY"
HOME	"HOMELKEY"
CTRL-HOME	"HOMEUKEY"
END	"HOMERKEY"
CTRL-END	"HOMEDKEY"
UP ARROW	"CURSUKEY"
CTRL-UP ARROW	"ROLLUKEY"
DOWN ARROW	"CURSDKEY"
CTRL-DOWN ARROW	"ROLLDKEY"
RIGHT ARROW	"CURSRKEY"
LEFT ARROW	"CURSLKEY"
PAGE DOWN	"NEXTKEY"
PAGE UP	"PREVKEY"
INSERT	"INSCKEY"
BACKSPACE	"BSKEY"
F1 through F10	"F1KEY" through "F10KEY"

RELATED COMMANDS

To map a PC key so that it performs the function of some other PC key, use MAPKEY.

LOAD

The LOAD command loads the specified configuration file.

SYNTAX

```
LOAD fname
```

fname
The name of the configuration file to be loaded.

EXAMPLE

```
LOAD DEFAULT.W92
```

In the above example, the configuration file DEFAULT.W92 is loaded into WS92.

RELATED COMMANDS

To save configuration file settings, use the SAVE command.

LOCF

The LOCF command names a file on the PC for file transfer.

When uploading, this is the file being transferred to the host.

When downloading, this is what the host file will be called on the PC.

SYNTAX

```
LOCF fname
```

fname
The name of the local file.

EXAMPLE

```
LOCF README.TXT
HOSTF MS92305.README.MINISOFT
ASCII
DOWNLOAD
```

The above example transfers the host file MS92305.README.MINISOFT to the PC, where it will be called README.TXT in the current directory.

```
LOCF C:\DATA\BUDGET.DAT
HOSTF BUDGET
BINARY
RECSIZE 256
UPLOAD
```

This example transfers the local file C:\DATA\BUDGET.DAT to the host, where it will be called BUDGET in the user's logon group and account.

RELATED COMMANDS

To define the name of a host file for file transfer, use HOSTF.

The commands to transfer files are DOWNLOAD, RECEIVE, and UPLOAD.

LOG

The LOG command sends incoming data to a file and/or printer.

SYNTAX

```
LOG [OFF]
```

OFF

Using LOG with no parameters turns logging on. Using LOG with the OFF parameter turns logging off.

EXAMPLE

```
CLOSE PRINTER
OPEN FILELIST DELETE
LOG
SEND "LISTF"
WAITC 17
LOG OFF
CLOSE DISK
```

This example captures the host's response to a LISTF command to a file named FILELIST in the current directory on the PC.

RELATED COMMANDS

To close an open file or the current "to" device, use the CLOSE command.

To open a file or device, use the OPEN command.

LOGCOLS

SYNTAX

LOGCOLS [N]

n being the number to change width of columns.

EXAMPLE

The below example demonstrates how a configuration file can be loaded and the number of rows and columns changed. The configuration file is then saved to a different file name.

```
LOAD DEFAULT.W92
LOGCOLS 149
LOGROWS 46
SAVE LOGCOLS.W92
END
```

LOGROWS

SYNTAX

LOGROWS [n]

n being the number to change the number of rows.

EXAMPLE

The below example demonstrates how a configuration file can be loaded and the number of rows and columns changed. The configuration file is then saved to a different file name.

```
LOAD DEFAULT.W92
LOGCOLS 149
LOGROWS 46
SAVE LOGCOLS.W92
END
```

LOWER

The LOWER command changes all uppercase characters in a specified string to lowercase.

SYNTAX

```
LOWER (string)
```

string
Variable name or string delimited by quotation marks. You may use the '&' operator to concatenate (join) strings.

EXAMPLE

```
LET Var1 = "Minisoft"
LET Var2 = LOWER(Var1)
TELL Var2
END
```

In the above example, a variable named Var1 is defined as the string "Minisoft", while a variable named Var2 uses LOWER to change the uppercase characters to lowercase. The contents of Var2, "minisoft", will then be displayed in a dialog box by the TELL command.

RELATED FUNCTIONS

UPPER

MAINVER

The MAINVER command is a predefined integer variable that returns the current main version number of Minisoft 92 (For example, MAINVER of version 5.2.42 is 5).

SYNTAX

```
MAINVER
```

EXAMPLE

```
LET Var1 = STRING(MAINVER)
LET Var2 = STRING(MIDVER)
LET Var3 = STRING(SUBVER)
LET Var4 = "The current version is: " & Var1 & "." & Var2 & "." &
Var3
TELL Var4
END
```

The above example sets Var1 to the string value of the main version number, Var2 mid version number, and Var 3 to the sub version number. Var4 is set to the values of the three numbers making up the version with embedded periods. This is then displayed in a dialog box.

RELATED FUNCTIONS

MIDVER
SUBVER

MCLEAR

The MCLEAR command homes and clears memory.

SYNTAX

MCLEAR

MDCMD

The MDCMD command is a machine dependent command. It controls internal emulator functions according to the table below:

MDCMD_COPYALL	Copies all display memory to the Window Clipboard.
MDCMD_PASTE	Pastes text from the Windows Clipboard to the input buffer, as if it were typed by the user.
MDCMD_NEXTFKSET or MDCMD_PREVFKSET	Cycles through HP function keys (USER, SYSTEM, MODES).
MDCMD_REFRESH	Calls the Windows routine for redrawing the WS92 window.
MDCMD_MAXIMIZE	Maximizes the screen.
MDCMD_MINIMIZE	Minimizes the task bar.
MDCMD_RESTORE	Restores the screen from the task bar.

MID

The MID command returns the characters in a specified string, between and including a specified beginning and ending point.

SYNTAX

```
MID (string,start,end)
```

string
Variable name or string delimited by quotation marks.

start
Defines start point as numeric constant or function.

end
Defines end point as numeric constant or function.

EXAMPLE

```
LET VAR1 = "PROG.PUB.SYS"
LET VAR2 = MID(VAR1, 1, FIND("SYS", VAR1)-2)
TELL VAR2
END
```

In the above example, the start parameter is the first character of the string "PROG.PUB.SYS". The end parameter is the result of the FIND function, which is the eighth character of the string. The resulting string, "PROG.PUB" will be defined as the variable VAR2.

RELATED FUNCTIONS

FIND
LENGTH

MIDVER

The MIDVER command is a predefined variable that returns the current mid-version number of Minisoft 92 (For example, the MIDVER of version 5.2.42 is 2).

SYNTAX

```
MIDVER
```

EXAMPLE

```
LET Var1 = STRING(MAINVER)
LET Var2 = STRING(MIDVER)
LET Var3 = STRING(SUBVER)
LET Var4 = "The current version is: " & Var1 & "." & Var2 & "." &
Var3
TELL Var4
END
```

The above example sets Var1 to the string value of the main version number, Var2 mid version number, and Var 3 to the sub version number. Var4 is set to the values of the three numbers making up the version with embedded periods. This will then be displayed in a dialog box.

RELATED FUNCTIONS

MAINVER
SUBVER

MICMD

The MICMD command is a machine independent command. It controls internal emulator functions according to the Machine Independent Command table.

SYNTAX

MICMD *command*

command
See one of the commands from the list of Machine Independent Commands listed in the following table.

EXAMPLE

```
MICMD MICMD_REDRAW
MICMD MICMD_UKREST
MICMD MICMD_QUIT
SEND BYE
END
```

The above example runs the *MICMD_REDRAW, UKREST,* and *QUIT* commands.

MACHINE INDEPENDENT COMMANDS

MICMD_FKTOG	Function Key toggle.
MICMD_UKREST	Restores user keys.
MICMD_REDRAW	Redraws text area of screen.
MICMD_QUIT	Quits the emulator.
MICMD_HOMELEFT	Moves cursor position to the first column.
MICMD_HOMERIGHT	Moves cursor to the last character of the current line.
MICMD_TYPEAHEAD	Toggles type-ahead feature.
MICMD_PRINTPAGE	Prints the current page.

MICMD_BREAK	Sends a BREAK to the host.
MICMD_DISC	Drops the connection mainly used with LAN connectivity.
MICMD_STOP	Toggles option that acts like a pause. The first one will stop a display and the scond will start the display (toggle).
MICMD_PRTFF	Sends a form feed to the printer.
MICMD_PRTCLOSE	Spools the file, like time out. Not the same as PRINT CLOSE.
MICMD_80	Sets the number of screen columns to 80.
MICMD_132	Sets the number of screen columns to 132.
MICMD_200	Sets the number of screen columns to 200.
MICMD_LOGBTOG	Log Bottom toggle.

NCONNECT

SYNTAX

NCONNECT [hostname]

EXAMPLE

```
LOAD DEFAULT.W92
NCONNECT “SUPPORT”
SAVE SUPPORT.W92
END
```

This example loads the DEFAULT.W92 configuration file, changes the NSVT host name, and saves the config file as SUPPORT.W92.

NEXTC

The NEXTC command waits for the next character to be received or until the time specified in the last TIMER command has elapsed with no characters being received.

SYNTAX

```
NEXTC
```

EXAMPLE

```
XMITC 13
NEXTC
DISPLAY “This is the NEXTC command”
SEND BYE
HARDEXIT
END
```

In the above example, the NEXTC command is waiting for a response to sending a carriage return.

RELATED COMMANDS

TIMER
ONTIMER
WAITC

NOBREAK

The NOBREAK command prevents a user from interrupting the execution of a script.

SYNTAX

```
NOBREAK
```

EXAMPLE

```
NOBREAK
WAITC 17
LOCF C:\DATA\BUDGET.DAT
HOSTF BUDGET.DATA.MINISOFT
RECSIZE 256
BINARY
UPLOAD
```

The above example does not allow the user to interrupt the script during a file transfer.

ONTIMER

The ONTIMER command causes execution of the script to jump to a specified line when the time specified by the TIMER command elapses.

Remember that the TIMER starts only upon execution of a NEXTC, WAITC, or WAITS command.

SYNTAX

```
ONTIMER lab
```

lab
Label for the line to which the execution is to jump.

EXAMPLE

```
LABEL CR
XMITC 13
TIMER 5
ONTIMER CR
WAITC 17
```

In the above example, execution jumps back to the line labeled CR if 5 seconds expire before a system prompt is received from the host (WAITC 17). This example is a loop that sends a carriage return (XMITC 13) if a system prompt is not received within 5 seconds of the previous carriage return.

RELATED COMMANDS

To set the timer, use the TIMER command.

To label a line where execution of the script is to jump, use a colon (:) or the LABEL command.

To start the timer by waiting for the next character from the host, use the NEXTC command.

To start the timer by waiting for a specific ASCII code from the host, use the WAITC command.

To start the timer by waiting for a specific string from the host, use the WAITS command.

OPEN

The OPEN command opens a PC file for read/write access. There maybe up to five files opened at once.

SYNTAX

To open a log file:

```
OPEN fname [APPEND | DELETE] [ASCII | BINARY]
```

To open a file for read/write access:

```
OPEN fname {INPUT | OUTPUT | APPEND | DELETE} AS n
[ASCII | BINARY]
```

fname
The name of a file or device to be opened.

APPEND
If the file is an existing file, use the APPEND option to write to the end of the existing file. For a log file or device, this parameter is optional. For a read/write access file, you must specify APPEND, DELETE, INPUT, or OUTPUT.

DELETE
Use this option to overwrite an existing file. For a log file or device, this parameter is optional. For a read/write access file, you must specify APPEND, DELETE, INPUT, or OUTPUT.

ASCII
Use the ASCII option to read and write to the file in ASCII mode. This parameter is optional.

BINARY
Use the BINARY option to read and write to the file in BINARY mode. This parameter is optional.

INPUT
Use the INPUT option if the file will be read from. For a read/write access file you must specify APPEND, DELETE, INPUT, or OUTPUT.

OUTPUT

Use the OUTPUT option if the file will be written to. For a read/ write access file you must specify APPEND, DELETE, INPUT, or OUTPUT.

n

Specifies file number which is used in READ, WRITE, and CLOSE commands for this file. This number must be in the range of 1-5.

EXAMPLE

```
CLOSE 3
OPEN TEXTFILE INPUT AS 3
READ 3 Var1
CLOSE 3
```

The above example opens the PC file TEXTFILE as a data source to be read from, and assigns it a file number of 3. The script then pauses 1 second and reads data from the file to a variable called Var1. The script then closes the file.

```
CLOSE PRINTER
OPEN FILELIST DELETE
LOG
SEND "LISTF"
WAITC 17
LOG OFF
CLOSE DISK
```

The above example captures the host's response to a LISTF command to a file named FILELIST in the current directory on the PC.

RELATED COMMANDS

To close a file, use the CLOSE command.

To read from a file, use the READ command.

To write to a file, use the WRITE command.

PRINTBY

The PRINTBY command sets the printer driver.

SYNTAX

PRINTBY {WINDOWS, PASSTHRU}

QUIT

The QUIT command exits Minisoft 92 and sets the DOS error level if communicating over a Serial Port. The user will remain logged on to the host, but if they are a Network user they are disconnected.

SYNTAX

```
QUIT [n]
```

n

DOS error level. This parameter is optional; default is 0.

EXAMPLE

```
SEND BYE
QUIT
```

This example logs the user off the host and exits from Minisoft 92.

RELATED COMMANDS

To terminate the connection between the PC and host, use the DISCONNECT command.

To send a break signal to the host, use the BREAK command.

To exit out of Minisoft 92, use the EXIT or HARDEXIT commands.

READ

The READ command reads from a specified data file to a variable such as linefeed.

SYNTAX

```
READ n variable
```

n

Specifies the file number used to open this file. Must be in the range of 1-5.

variable

Name of a variable that will store what is read from the file. The variable will store up to 80 characters.

EXAMPLE

```
CLOSE 3
OPEN TEXTFILE INPUT AS 3
READ 3 Var1
CLOSE 3
```

The above example opens the PC file TEXTFILE as a source of data to be read from, and assigns it a file number of 3. The script then reads data from the file to a variable called Var1. The script then closes the file.

RELATED COMMANDS

To *close* a file, use the CLOSE command.

To *open* a file, use the OPEN command.

To *write* to a file, use the WRITE command.

READHOST

The READHOST command reads host output into a variable. Output will be read until a carriage return is reached, a time limit, or character other than carriage return is specified to end the command.

SYNTAX

READHOST [time] variable1 [UNTIL string] [LIMIT n] [TERMINATOR variable2]

time
Amount of time to wait for host output before canceling the READHOST command. Format is HH:MM:SS. This parameter is optional.

variable1
The name of the variable where the host output is to be stored. The variable will store up to 80 characters.

UNTIL *string*
A character used, instead of a carriage return to end the READHOST command. Specifying more than one character does not define a termination string for the command. Rather, each of the characters acts as a terminator. This parameter is optional. READHOST will terminate at a carriage return (^M) by default.

LIMIT *n*
The number of characters to be read, if fewer than 80. This parameter is optional.

TERMINATOR *variable2*
The name of a variable to store the character that terminates the READHOST command. If time is exceeded, the length of this variable will be 0.

EXAMPLE

```
READHOST 0:05:00 Var1 UNTIL "^J" TERMINATOR Var2
```

In the above example, the script will wait 5 minutes for host output, which it stores in variable Var1. It will read host output until it receives a linefeed and returns a linefeed character, if it receives one, as the value of the variable Var2. If the command times out, the length at Var2 will be 0.

RELATED COMMANDS

To read data from a file to a variable, use the READ command.

RECEIVE

The RECEIVE command transfers a file from the host to the PC.

R acts as RECEIVE.

SYNTAX

```
RECEIVE LOCF FROM HOSTF [ASCII | BINARY]
```

LOCF
Name of the file that will be on the PC.

HOSTF
Name of the file being downloaded from the host.

ASCII
Denotes file transfer as ASCII (text mode). If you do not specify ASCII, the file will transfer as binary.

BINARY
Denotes file transfer as binary (binary image). This is the default.

EXAMPLE 1

```
RECEIVE SALES.RPT FROM SALESRPT BINARY
```

The above example transfers the host file SALESRPT to the PC, where it will be called SALES.RPT in the current directory. The transfer is binary.

EXAMPLE 2

The RECEIVE command accepts variables:

```
LET V1=FILE1
LET V2=NPREADME.PUB
RECEIVE V1 FROM V2 ASCII
END
```

RELATED COMMANDS

Before using this command, you must set a record size using the RECSIZE command.

If LOCF and HOSTF have already been defined, use the DOWNLOAD command.

RECSIZE

The RECSIZE command is used during file transfer. Record size in the specified number of bytes.

SYNTAX

```
RECSIZE n
```

n

The number of bytes per record.

EXAMPLE

```
LOCF C:\WINWORD\README.DOC
HOSTF MS92305.README.MINISOFT
BINARY
RECSIZE 256
UPLOAD
```

The above example uploads a binary file. The host file's record length will be 256.

```
LOCF README.TXT
HOSTF MYFILE.TEXT.MINISOFT
ASCII
RECSIZE 80
UPLOAD
```

The above example uploads an ASCII file. The host file's record length will be 80.

RELATED COMMANDS

You must specify a record size when using the UPLOAD command. Also, the RECSIZE command must precede the UPLOAD command.

RETURN

The RETURN command returns to normal execution from a subroutine specified by the last GOSUB command.

SYNTAX

```
RETURN
```

EXAMPLE

```
GOSUB CLEAR
        .
        .
        .
:CLEAR
DISPLAY "^[H^[J"
RETURN
```

In the above example, a GOSUB command runs the subroutine that begins with the line labeled CLEAR. The subroutine sends the cursor to the upper left corner of the screen and clears the display, then returns execution to the script with the RETURN command.

RELATED COMMANDS

You must end a subroutine specified by GOSUB with the RETURN command.

RUN

The RUN command runs a DOS shell or Windows Program Manager (CTRL-F10) and a DOS program, if one is specified. The script file continues to execute while the specified program is running.

SYNTAX

```
RUN [cmd]
```

cmd

A command to execute the desired DOS or Windows program. If a command is not specified, the DOS shell or DOSPRMPT.PIF is executed.

EXAMPLE

```
SHELL EDIT.COM
```

In the above example, the script runs the DOS editor. The script does not pause execution while the editor is running.

RELATED COMMANDS

To run a DOS shell that causes the script to pause until the DOS program is finished, use the SHELL command with the "nowait" parameter.

S

The S command sends a file to the Host.

SYNTAX

S <PCFile> to <Hostfile> [ASCII] [Binary] [Resize] [Delete]

SAVE

The SAVE command keeps the current configuration settings to a local file with the specified name.

SYNTAX

```
SAVE fname
```

fname
Name for the configuration file.

EXAMPLE

```
SAVE MS92.CFG
```

In the above example, the current configuration settings are saved to the file MS92.CFG in the user's current PC directory. MS92.CFG is the default configuration file for DOS92.

```
SAVE unixlan.W92
```

In the above example, the current configuration settings are saved to the file UNIXLAN.W92 in the user's current PC directory. The .W92 extension is the proper syntax for configuration files in WS92.

RELATED COMMANDS

To run a particular set of configuration settings, use the LOAD command.

SAVINF

The SAVINF command saves file header information from a host file (record size, block size, etc.) when downloading a file. This information is saved in the first 128 bytes of the local file and is useful if you want to upload a binary file to another host or to the same host under a different file name.

SYNTAX

```
SAVINF
```

EXAMPLE

```
LOCF LINKFILE
HOSTF MS92LNK4.PUB.MINISOFT
BINARY
SAVINF
DOWNLOAD
```

In the above example, a binary file is downloaded to the PC with its file header information saved. The file's attributes are then preserved in case the file is re-uploaded to the host, as in the following example:

```
LOCF LINKFILE
HOSTF TESTLINK.MYGROUP.MINISOFT
BINARY
UPLOAD
```

SCROLLBAR

The SCROLLBAR command sets right scroll bar.

SYNTAX

SCROLLBAR {ALWAYS, UNMAXED, NEVER}

SEMICOLON

Any command line that begins with a SEMICOLON is treated as a comment line and is not executed. Do not place commands that you wish to be executed in a comment line.

SYNTAX

;

Text that you do not want the script file to execute.

EXAMPLE

```
;************************************************
; Convert $TIME to 12 hour format of HH:MM AM/PM
; Uses the VALUE function to convert the format
;************************************************
LET HH = MID($TIME,1,2)
LET MM = MID($TIME,4,5)
LET TEMPHH = VALUE(HH)
IF TEMPHH > 12
  LET TEMPHH = TEMPHH - 12
  LET HH = STRING(TEMPHH)
  LET PM = "PM"
ELSE
  LET PM = "AM"
ENDIF
LET PCTIME = HH & ":" & MM & " " & PM
;************************************************
; Set up display to show both formats
;************************************************
LET HOLD = "$TIME in a 24hr display " & $TIME & "^M"
LET HOLD = HOLD & "Time converted to 12hr display: " &
PCTIME
TELL HOLD
END
```

In the above example, the comment line describes the action of the script lines that follows.

SEND

The SEND command transmits a string, followed by a carriage return.

SYNTAX

```
SEND string
```

string
A literal string, not delimited by quotation marks.

EXAMPLE

```
SEND HELLO MGR.MINISOFT
```

The above example transmits a logon to an HP e3000 host.

RELATED COMMANDS

To transmit a string without sending a carriage return, use TRANSMIT.

SET

The SET command sets configuration options.

SYNTAX

```
SET DISABLE-COMP-CODES [yes | no]
```

NO
: The S and F result codes are sent to the host in response to a host-initiated command—i.e., a command beginning with the escape sequence esc&oC (see *Host-initiated commands* in *Appendix C*).

YES
: The S and F result codes are not sent to the host in response to a host-initiated command.

EXAMPLE

```
SET RIGHT-MARGIN #
END
```

#
: The number of the column at which the text will wrap to the next line.

```
SET DISPLAY-ROWS nn
```

nn
: Sets the number of rows in the display to the value of nn.

```
SET TERMINAL-TYPE HP
```

Changes the actual emulation.

```
SET TERMINAL-TYPE HP2329A
```

Changes the type reported to Telnet on login.

Note: You can set terminal-type to other strings and it will report that type when making a telnet connection, but will not change the emulation internally unless it matches one of the above. For example if you wish to be in HP emulation but the host recognizes “HP2392A”, you can do the command twice:

SET WINDOW-TITLE <title name>

SHELL

The SHELL command runs a DOS shell or Windows Program Manager (CTRL-F10) and a DOS program if one is specified (ALT-F10).

SYNTAX

```
SHELL [cmd [NOWAIT]]
```

cmd

A command to execute the desired DOS or Windows program. If a command is not specified, the DOS shell or DOSPRMPT.PIF is executed.

NOWAIT

Causes the script file to continue execution while the specified program is running. The default behavior of the SHELL command with a DOS command parameter is to pause execution of the script file until the DOS program terminates.

EXAMPLES

```
SHELL SORT.EXE MS92FILE > SORTFILE NOWAIT
```

In the above example, the script runs a DOS shell that performs a DOS sort. The script continues execution while the sort is running.

```
SHELL EDIT.COM
```

In the above example, the script runs the DOS editor. The script pauses execution until the user exits from the editor.

RELATED COMMANDS

To run a DOS shell that does not need the "nowait" parameter to leave script execution uninterrupted, use the RUN command.

STOP

The STOP command stops the execution of a script file.

SYNTAX

```
STOP
```

EXAMPLE

```
ASK "Halt command? Yes/No:"
IFYES STOPIT
  .
  .
  .
LABEL STOPIT
STOP
```

In the above example, the script executes a STOP command on the condition of a user responding "yes" to the question "Halt command?"

RELATED COMMANDS

To mark the end of the script file, use the END command.

TABLOAD

The TABLOAD command loads alternate translation tables without having to exit MS92.

SYNTAX

```
TABLOAD type fname
```

type
Indicates the type of character set translation.

fname
Indicates the translation table to load.

In the table below, the third column lists the translation tables automatically loaded (if they are present) when MS92 is started. The Tabload command lets you load a different translation table of a selected type to perform the indicated function.

Function	Translation Table	TABLOAD Value
PC keyboard to host computer	HP_CHARS.TBL	1
Host computer to PC monitor	PC_CHARS.TBL	2
ASCII file transfer from PC to host	XLAT1.TBL	3
ASCII file transfer from host to PC, and capture to disk	XLAT2.TBL	4
Host computer to PC monitor; for function key labels	FK_CHARS.TBL	5
Host computer to PC slaved printer (local print functions)	PT_CHARS.TBL	6
Copy-and-paste from WS92 to other Windows applications	CUTCHARS.TBL	7
Paste from other Windows applications to WS92	PASTE_CH.TBL	8

EXAMPLE

```
TABLOAD 1 PC8SWE7.TBL
```

In the above example, the file PC8SWE7.TBL is being loaded to translate keyboard input characters to the host character set. If the HP_CHARS.TBL table exists, PC8SWE7.TBL will now take its place for the remainder of the current session.

TCONNECT

The TCONNECT command makes a TELNET connection to the host with a node name or IP address.

SYNTAX

```
TCONNECT [hostname]
```

EXAMPLE

```
LOAD DEFAULT.W92
HOSTPORT 23
TCONNECT “209.23.116.12”
SAVE SUPPORT.W92
END
```

The above example demonstrates how a configuration file can be loaded and the node name or IP address specified. The port number must also be defined by the HOSTPORT command.

TELL

The TELL command presents a message to the user, and waits for the user to press a key.

SYNTAX

```
TELL "string"
```

"*string*"
Text to be displayed to the user, delimited by quotation marks. You may use the '&' operator to concatenate (join) strings.

EXAMPLE

```
LET Var1 = "Press a key:"
TELL "Could not connect to host. " & Var1
```

The above example prints the message *"Could not connect to host. Press a key:"* to the user and pauses execution of the script until the user presses a key.

RELATED COMMANDS

To ask the user a yes/no question, use the ASK command.

TIMER

The TIMER command sets the timer for a specified number of seconds. The timer starts upon the execution of the next WAITC, WAITS, or NEXTC command.

SYNTAX

```
TIMER val
```

val
Number of seconds.

EXAMPLE

```
TIMER 40
ONTIMER NOANSWER
WAITS CONNECT 9600
```

In the above example, the timer has been set for 40 seconds. If the string, "CONNECT 9600" is not received before the amount of time elapses, execution of the script will jump to the line labeled NOANSWER.

RELATED COMMANDS

To execute script commands due to the timer's lapse, use the ONTIMER command.

TRACE

The TRACE command displays script commands as they are executed.

SYNTAX

```
TRACE
```

EXAMPLE

```
TRACE
SEND ATDT15551212
TIMER 40
ONTIMER NOANSWER
WAITS CONNECT 9600
```

In the above example, all script commands following the TRACE command are displayed to the screen as they are executed.

TRANSMIT

The TRANSMIT command sends data to the host without sending a carriage return. The data maybe a string in quotation marks or the contents of a specified variable.

SYNTAX

```
TRANSMIT "string" | variable
```

"*string*"
Transmits the string in quotation marks. To include a control character in the string, precede the character with a caret (^), such as ^J for linefeed. You may use the '&' operator to concatenate (join) strings.

variable
Transmits the contents of the specified variable.

EXAMPLES

```
TRANSMIT "hello mgr.minisoft,"
```

The above example transmits a logon of the user and account name, but does not transmit a carriage return, allowing the user to supply a group name at run time.

RELATED COMMANDS

To transmit a string followed by a carriage return, use the SEND command.

UPLOAD

The UPLOAD command transfers a file from the PC to the host.

SYNTAX

```
UPLOAD
```

EXAMPLE

```
LOCF C:\WINWORD\README.DOC
HOSTF MS92305.README.MINISOFT
BINARY
RECSIZE 256
UPLOAD
```

The above example transfers the PC file C:\WINWORD\README.DOC to the host, where it will be called MS92305.README.MINISOFT. Since the transfer mode is BINARY, the host file will be a binary image of the PC file. The host file's record length will be 256.

RELATED COMMANDS

To use the UPLOAD command, you must have already defined LOCF and HOSTF in the script. You must also specify ASCII or BINARY, as well as RECSIZE before issuing the UPLOAD command.

Use the APPEND command with the UPLOAD command to append data to the end of an existing file. For example:

```
LOCF C:\WINWORD\README.TXT
HOSTF MS92305.README.MINISOFT
APPEND
ASCII
RECSIZE
UPLOAD
```

WAIT

The WAIT command causes execution of a script to pause until a specified time of day, specified amount of time for a particular string, or for silence from the host.

SYNTAX 1

```
WAIT [UNTIL] time [FOR string]
```

time

Time in the format of HH:MM:SS.

With the UNTIL option, this is the time of day. If the UNTIL option is not used, this is the amount of time to wait before resuming execution of the script.

FOR *string*

String to be received from the host. Maybe a named variable or a string delimited by quotation marks. You may use the '&' operator to concatenate (join) strings.

SYNTAX 2

```
WAIT FOR time SILENCE
```

Specifies an amount of time during which no data is received from the host. Time is in the format of HH:MM:SS.

EXAMPLES

```
WAIT 0:0:10 FOR "PASSWORD"
```

In the above example, the script will wait 10 seconds to receive the string "PASSWORD" before proceeding.

```
WAIT FOR 00:01:00 SILENCE
TELL "Host not responding. Press a key."
```

In the above example, the script waits for 1 minute of silence from the host before executing a TELL command to the user.

PREDEFINED VARIABLE

The predefined variable FOUND is updated after a timed WAIT command.

EXAMPLE

```
WAIT 0:00:45 for "abc"
IF FOUND
   DISPLAY "abc was received before 45 seconds
        had elapsed."
ENDIF
```

RELATED COMMANDS

To cause the script to wait for a particular ASCII character, use the WAITC command.

To cause the script to wait for a particular string (without specifying an amount of time), use the WAITS command.

WAITC

The WAITC command waits until a specified character is received or until the time specified in the last TIMER command has elapsed with no characters being received.

SYNTAX

```
WAITC c
```

c

Character to be received by NEXTC, specified by ASCII code.

EXAMPLE

```
WAITC 17
```

In the above example, the script waits for a system prompt (ASCII 17 = ^Q) before resuming execution.

RELATED COMMANDS

To cause the script to wait for a particular string, use the WAITS command.

To set a timer for the response to the WAITC command, use the TIMER command.

To cause execution of the script to jump to a specific line when the timer expires, use the ONTIMER command.

WAITS

The WAITS commandWaits until the specified string is received or until the time specified in the last TIMER command has elapsed with no characters being received.

SYNTAX

```
WAITS string
```

string
A literal string, not delimited by quotation marks.

EXAMPLE

```
WAITS ^Q
```

In the above example, the script is waiting for a system prompt before resuming execution.

RELATED COMMANDS

To cause the script to wait for a particular ASCII character, use the WAITC command.

To set a timer for the response to the WAITC command, use the TIMER command.

To cause execution of the script to jump to a specific line when the timer expires, use the ONTIMER command.

WINTITLE

The WINTITLE command changes main window title text.

EXAMPLE

```
LET V1=ABC
WINTITLE V1
END
```

or

```
WINTITLE ABC
END
```

WRITE

The WRITE command writes a string to an opened file.

SYNTAX

```
WRITE n string
```

n
Specifies file number used to open this file. Must be in the range of 1-5.

string
A literal string, not delimited by quotation marks.

EXAMPLE

```
CLOSE 3
OPEN TEXTFILE OUTPUT AS 3
WRITE 3 Var1
CLOSE 3
```

The above example opens the PC file TEXTFILE as a file that can be written to and assigns it a file number of 3. The script then pauses 1 second and writes data from a variable called Var1 to the file. The script then closes the file.

RELATED COMMANDS

To close a file, use the CLOSE command.

To open an existing file, use the OPEN command.

To read from a file, use the READ command.

XMITC

The XMITC command transmits the specified ASCII character code to the host without adding a carriage return.

SYNTAX

```
XMITC c
```

c
ASCII code to be transmitted.

EXAMPLE

```
XMITC 10
```

The above example sends a linefeed to the host.

RELATED COMMANDS

To transmit strings of text characters and control codes, use the TRANSMIT or XMITS commands.

XMITS

The XMITS command transmits the specified string without adding a carriage return.

SYNTAX

```
XMITS string
```

string
A literal string, not delimited by quotation marks.

EXAMPLE

```
XMITS ^J
```

The above example sends a linefeed to the host.

RELATED COMMANDS

To transmit a string with delimiters of quotation marks, use the TRANSMIT command.

To transmit a particular ASCII code, use the XMITC command.

DDE COMMANDS

The DDE commands apply only to WS92.

WS92 supports the message protocol designed by Microsoft for DDE (Dynamic Data Exchange). The commands in this protocol allow separate Windows applications to establish links whereby they may share the same data. For example, a Windows spreadsheet application linked to the host through WS92 could automatically update its information as changes are made to data on the host.

In DDE, the application that seeks to access data (client application) initiates communication (conversation) with the application in which the data originates (server application). DDE commands establish links and control communication between the applications, allowing data in the client application to be updated automatically by the server application.

WS92's Service Name is configured on the DDE Configuration menu. WS92's Topic Name is always S92.

DDE must be enabled in the DDE Configuration dialog box for WS92 to act as either a DDE server or client (see the DDE section of *Chapter 2* for more information on configuring for DDE).

WS92 may act as either a DDE server or DDE client application. As a client, WS92 issues any of the commands listed here. As a server, WS92 responds to DDE-ADVISE, DDE-EXECUTE, DDE-POKE, DDE-REQUEST, and DDE-UNADVISE commands issued from a client.

Following is an alphabetical reference of DDE client commands.

DDE-EXECUTE

The DDE-EXECUTE command causes the server application to execute one or more specified commands in its own script or macro language.

SYNTAX

DDE-EXECUTE <conversation num> <command string>

The <*conversation num*> is the value returned by an earlier DDE-INITIATE command. The <*command string*> uses the DDE standard command syntax. Square brackets delimit each command.

EXAMPLE

The following example assumes a conversation number V0 was initiated with Excel, naming a worksheet (such as BUDGET.XLS) as the topic. The DDE-EXECUTE command causes Excel to scroll the worksheet to row 50, using Excel's VSCROLL command:

DDE-EXECUTE V0 "[VSCROLL(50, TRUE)]"

DDE-INITIATE

The DDE-INITIATE command starts a DDE conversation between WS92 (as the DDE client) and the specified application (as the DDE server). The specified conversation topic must be supported by the server application. The conversation number (an integer from 0 to 24) is stored in the specified variable. This conversation number is used to identify the conversation in subsequent DDE client commands. A DDE conversation is specified by an application name and a topic. If more than one DDE server application responds (see the discussion of wildcards below), a conversation is initiated with only the first server responding. The server application's user manual should contain descriptions of the DDE topics supported by that application.

SYNTAX

```
DDE-INITIATE <application> <topic> <var>
```

The <*application*> is a string expression that corresponds to a DDE server application name. An empty string ("") may be used for this parameter and is treated as a wildcard to find all DDE server applications with the specified <*topic*>. The <*topic*> is a string expression that corresponds to the desired DDE conversation topic. An empty string ("") may be used for this parameter and is treated as a wildcard to find the DDE conversation topics supported by the specified <*application*>. The <*var*> specifies a variable for the conversation number.

EXAMPLE

The following example issues a command that causes WS92 to initiate a DDE conversation with Excel, with a topic of BUDGET.XLS, allowing WS92 to exchange data with the worksheet named BUDGET.XLS:

```
DDE-INITIATE "EXCEL" "BUDGET.XLS" V0
```

DDE-NAME

The DDE-NAME command changes DDE Service Name.

SYNTAX

DDE-NAME (NAME)

DDE-POKE

The DDE-POKE command sends the item value to the named item in the server application of the specified conversation. The effect of this command is to send the server's item to a specified value.

SYNTAX

DDE-POKE <conversation num> <item name> <item val>

The <*conversation num*> is the value returned by an earlier DDE-INITIATE command. The <*item name*> is a string expression telling the server what data is being sent. The <*item val*> is a string expression containing the data to send to the server. For valid data items, see the DDE server application's user manual.

EXAMPLE

The following example assumes a conversation number V0 was initiated with Excel, naming a worksheet (such as BUDGET.XLS) as the topic. The command puts a value of 33.44 in a cell at row 50, column 5 of the worksheet:

DDE-POKE V0 "R50C5" "33.44"

DDE-REQUEST

The DDE-REQUEST command requests an item name from the server application in the specified conversation and stores the data item value in the specified variable. This data value is in string format, and is empty if the DDE-REQUEST fails.

SYNTAX

DDE-REQUEST <conversation num> <item name> <var>

The <*conversation num*> is the value returned by an earlier DDE-INITIATE command. The <*item name*> is a string expression telling the server what data is being requested. For valid data items, see the DDE server application's user manual. The <*var*> specifies a variable for the conversation number.

EXAMPLE

The following example assumes a conversation number V0 was initiated with Excel, naming a worksheet (such as BUDGET.XLS) as the topic. The DDE-REQUEST command retrieves the contents of the worksheet cell at row 10, column 4, and places the value in WS92 variable V1.

```
DDE-REQUEST V0 "R10C4" V1
```

RELATED COMMANDS

DDE-ADVISE

DDE-SUPPORT

The DDE-SUPPORT command enables DDE support.

SYNTAX

DDE-SUPPORT (ON/OFF)

DDE-TERMINATE

The DDE-TERMINATE command terminates the specified DDE conversation. If there are any DDE advise-links associated with the conversation, they are removed.

SYNTAX

```
DDE-TERMINATE <conversation num>
```

EXAMPLE

The following example assumes a conversation number V0 was initiated with Excel, and terminates that conversation:

```
DDE-TERMINATE V0
```

RELATED COMMANDS

DDE-TERMINATE-ALL

DDE-TERMINATE-ALL

The DDE-TERMINATE-ALL command terminates all current DDE conversations initiated by earlier DDE-INITIATE commands. If there are any DDE advise-links for these conversations, they are removed.

SYNTAX

```
DDE-TERMINATE-ALL
```

RELATED COMMANDS

DDE-TERMINATE

FUNCTIONS

$DATE

The $DATE function is a predefined variable that returns the current date, according to the PC's CPU clock.

SYNTAX

$DATE

The date is returned in the format of MM-DD-YYYY.

EXAMPLE

```
;*************************************************
; Convert $TIME to 12 hour format of HH:MM AM/PM
;
LET HH = MID($TIME,1,2)
LET MM = MID($TIME,4,5)
LET TEMPHH = VALUE(HH)
IF TEMPHH > 12
  LET TEMPHH = TEMPHH - 12
  LET HH = STRING(TEMPHH)
  LET PM = "PM"
ELSE
  LET PM = "AM"
ENDIF
LET PCTIME = HH & ":" & MM & " " & PM
;*************************************************
;*************************************************
; Ask for the HP's Time and format to  HH:MM AM/PM
SEND SHOWVAR HPTIMEF
WAITC 17
VARGET ROW
LET SROW = ROW -1
```

```
LET TEMPTIME = SCREENRECT(SROW,0,SROW,79)
LET SCOL = FIND(":",TEMPTIME)
LET SCOL = SCOL - 3
LET ECOL = SCOL + 8
LET HPTIME = MID(TEMPTIME,SCOL,ECOL)
;************************************************
;************************************************
; Ask for the HP Date and format into MMM DD YYYY
SEND SHOWVAR HPDATEF
WAITC 17
VARGET ROW
LET SROW = ROW -1
LET TEMPDATE = SCREENRECT(SROW,0,SROW,79)
LET SCOL = FIND(",",TEMPDATE)
LET SCOL = SCOL + 2
LET ECOL = SCOL + 12
LET HPDATE = MID(TEMPDATE,SCOL,ECOL)
;************************************************
;************************************************
; Combine the HP and PC date and time in one display
LET BOTH =  "The HP's time is: " & HPTIME & " on " & HPDATE
& "^M"
LET BOTH = BOTH & "The PC's time is: " & PCTIME & " on " &
$DATE
TELL BOTH
END
```

The above example gets the time and date of both the PC and HP and then displays both times in a dialog box for comparison.

RELATED FUNCTIONS

$TIME

$ONLINE

The $ONLINE function is a predefined variable that returns the status of the connection.

SYNTAX

```
$ONLINE
```

The value of 1 if a connection is establish, 0 if not connected

EXAMPLE

```
CLOSE-CONNECTION
NCONNECT JAVELIN
IF $ONLINE = 0
 TELL JAVELIN IS NOT RESPONDING WILL TRY SUPPORT
 NCONNECT "SUPPORT"
  IF $ONLINE = 0
   TELL Both Javelin and Support are not responding call IS for
help
    EXIT
  ENDIF
ENDIF
END
```

In the above example all connection are closed, then an NSVT connection to Javelin is attempted. If for some reason that connection is not establish, a message is generated telling the user that Javelin is not responding and that a connection to Support will be tried. If both connections fail, a message is displayed that both connection are not responding and call IS for help. WS92 will then be terminated.

RELATED COMMANDS

CLOSE-CONNECTION
NCONNECT
TCONNECT

$TIME

The $TIME function is a predefined variable for the current time.

SYNTAX

$TIME

The current time is returned in the format HH:MM:SS:CC, on a 24-hour clock.

EXAMPLE

```
;************************************************
; Convert $TIME to 12 hour format of HH:MM AM/PM
;
LET HH = MID($TIME,1,2)
LET MM = MID($TIME,4,5)
LET TEMPHH = VALUE(HH)
IF TEMPHH > 12
  LET TEMPHH = TEMPHH - 12
  LET HH = STRING(TEMPHH)
  LET PM = "PM"
ELSE
  LET PM = "AM"
ENDIF
LET PCTIME = HH & ":" & MM & " " & PM
;************************************************
;************************************************
; Ask for the HP's Time and format to  HH:MM AM/PM
SEND SHOWVAR HPTIMEF
WAITC 17
VARGET ROW
LET SROW = ROW -1
LET TEMPTIME = SCREENRECT(SROW,0,SROW,79)
LET SCOL = FIND(":",TEMPTIME)
LET SCOL = SCOL - 3
LET ECOL = SCOL + 8
LET HPTIME = MID(TEMPTIME,SCOL,ECOL)
;************************************************
```

```
;************************************************
; Ask for the HP Date and format into MMM DD YYYY
SEND SHOWVAR HPDATEF
WAITC 17
VARGET ROW
LET SROW = ROW -1
LET TEMPDATE = SCREENRECT(SROW,0,SROW,79)
LET SCOL = FIND(",",TEMPDATE)
LET SCOL = SCOL + 2
LET ECOL = SCOL + 12
LET HPDATE = MID(TEMPDATE,SCOL,ECOL)
;************************************************
;************************************************
; Combine the HP and PC date and time in one display
LET BOTH =  "The HP's time is: " & HPTIME & " on " & HPDATE
& "^M"
LET BOTH = BOTH & "The PC's time is: " & PCTIME & " on " &
$DATE
TELL BOTH
END
```

The above example reformats HP's time into a 12 hour HH:MM (A-P)M format. Along with the time, the date will then display in a dialog box for comparison.

RELATED FUNCTIONS

$DATE

EXIST

The EXIST function tests for the existence of a specified local file. The value is true if the file exists.

SYNTAX

```
EXIST (fname)
```

(fname)

The name of a PC file, in parentheses. The name can include wildcards.

EXAMPLE

```
IF EXIST (*.s92)
        TELL "There are *.s92 files"
 ELSE
  TELL "No *.S92 file were found"
 ENDIF
 END
```

The above example tests for the existence of any Minisoft 92 configuration files (files whose extension is .S92) in the current directory. After it has finished checking, it will then display a dialog box indicating if any files were found or not.

FIND

The FIND function returns the location of a string within another string. The value FIND returns to the location in the second string where the first string is found. Thus, if the first string is found beginning with the second character of the second string, FIND returns a value of 2.

If the first string is not found within the second string, FIND returns a value of 0.

SYNTAX

```
FIND (string1, string2)
```

string1
Named variable or string delimited by quotation marks.

string2
Named variable or string delimited by quotation marks.

EXAMPLE

```
LET V1 = "MGR.MINISOFT "
LET V2 = FIND(".",V1)
LET V2 = V2 - 1
LET V3 = MID(V1,1,V2)
LET V2 = V2 + 2
LET V4 = MID(V1,V2,LENGTH(V1))
TELL "BEFORE THE PERIOD WAS: " & V3
TELL "AFTER THE PERIOD WAS: " & V4
END
```

In the above example, FIND defines the value of variable V2 as 4. It then uses that value to compute the value before the period and the value after the period. Using the TELL command it then displays them in a dialog box.

RELATED FUNCTIONS

LENGTH
MID

FOUND

The FOUND function is a predefined variable that returns true if the string specified in the most recent WAIT or READHOST command was found.

SYNTAX

```
FOUND
```

EXAMPLE

```
SEND LISTF COB@
WAIT 0:0:8 FOR "COBT"
IF NOT FOUND
  TELL "No file starting with COBT was found "
ELSE
  TELL "File(s) starting with COBT were found "
ENDIF
END
```

In the above example, the script sends a LISTF looking for all files that start with COB in the current group on the HP e3000. It then waits for eight seconds to see if any of the files have COBT in the name. A dialog box will then display a message of whether it found or did not find files starting with COBT.

RELATED FUNCTIONS

WAIT
READHOST

LENGTH

The LENGTH function returns the number of characters in a specified string.

SYNTAX

```
LENGTH (string)
```

string

A variable or string delimited by quotation marks. To include a control character in the string, precede the character with a caret (^), such as ^J for linefeed. You may use the '&' operator to concatenate (join) strings.

EXAMPLE

```
LET Var1 = "Last Record"
LET Var2 = LENGTH(Var1)
LET Var3 = "The length of Var1 is: "
LET Var3 = Var3 & STRING(Var2)
TELL Var3
END
```

In the above example, a variable named Var2 is defined as the length of the string, "Last Record" which is 11. Var3 sent the string "The length of Var1 is: " and using the STRING function, the integer value of Var2 was concatenated to the string value of Var3. The TELL command will display a dialog box with the message "The length of Var1 is: 11"

RELATED FUNCTIONS

FIND
STRING
TELL

LOWER

The LOWER function changes all uppercase characters in a specified string to lowercase.

SYNTAX

```
LOWER (string)
```

string

A named variable or string delimited by quotation marks. You may use the '&' operator to concatenate (join) strings.

EXAMPLE

```
LET Var1 = 'MiniSoft"
LET Var2 = LOWER(Var1)
TELL Var2
END
```

In the above example, a variable named Var1 is defined as the string "MiniSoft" while a variable named Var2 uses LOWER to change the uppercase characters of the string to lowercase. The content of Var2 is the string "minisoft". This will then be displayed in a dialog box by the TELL command.

RELATED FUNCTIONS

UPPER

MAINVER

The MAINVER function is a predefined integer variable that returns the current main version number of Minisoft 92 (For example, the MAINVER of version 5.2.42 is 5).

SYNTAX

```
MAINVER
```

EXAMPLE

```
LET Var1 = STRING(MAINVER)
LET Var2 = STRING(MIDVER)
LET Var3 = STRING(SUBVER)
LET Var4 = "The current version is: " & Var1 & "." & Var2 & "." &
Var3
TELL Var4
END
```

The above example sets Var1 to the string value of the main version number, Var2 the Mid version number, and Var 3 to the Sub version number. Var4 is set to the values of the three numbers making up the version with embedded periods. This is then displayed in a dialog box.

RELATED FUNCTIONS

MIDVER
SUBVER

MID

The MID function returns the characters in a specified string, between and including a specified beginning and ending point.

SYNTAX

```
MID (string,start,end)
```

string
Named variable or string delimited by quotation marks.

start
Defines start point as numeric constant or function.

end
Defines end point as numeric constant or function.

EXAMPLE

```
LET VAR1 = "PROG.PUB.SYS"
LET VAR2 = MID(VAR1, 1, FIND("SYS", VAR1)-2)
TELL VAR2
END
```

In the above example, the start parameter of the string is the first character of the string "PROG.PUB.SYS". The end parameter is the result of the FIND function, which is the eighth character of the string. The resulting string, "PROG.PUB" will be defined as the variable VAR2.

RELATED FUNCTIONS

FIND
LENGTH

MIDVER

The MIDVER function is a predefined variable that returns the current mid-version number of Minisoft 92 (For example, the MIDVER of version 5.2.42 is 2).

SYNTAX

```
MIDVER
```

EXAMPLE

```
LET Var1 = STRING(MAINVER)
LET Var2 = STRING(MIDVER)
LET Var3 = STRING(SUBVER)
LET Var4 = "The current version is: " & Var1 & "." & Var2 & "." &
Var3
TELL Var4
END
```

The above example sets Var1 to the string value of the main version number, Var2 the Mid version number, and Var 3 to the Sub version number. Var4 is set to the values of the three numbers making up the version with embedded periods, which is then displayed in a dialog box.

RELATED FUNCTIONS

MAINVER
SUBVER

SCREENFIELD

The SCREENFIELD function searches for a specified string in screen memory, and returns a field of data following the specified string.

SYNTAX

```
SCREENFIELD (string[,length,startrow,startcol])
```

string
Named variable or string delimited by quotation marks.

length
Specifies the length of the field. This parameter is optional.

startrow
Specifies the row on which the field begins. This parameter is optional.

startcol
Specifies the column in which the field begins. This parameter is optional.

EXAMPLE

```
DISPLAY "^[H^[J"
SEND HELP FCOPY PARMS
WAITC 17
LET VAR1 = SCREENFIELD("Reference Manual")
SEND HELP EDITOR PARMS
WAITC 17
LET VAR2 = SCREENFIELD("file",13,18,10)
TELL "This is what follows Reference Manual in Help FCOPY
^mPARMS until the first cr/lf " & VAR1
TELL "This is what follows the first 'file' found after line 18^M
column 10 for the length of 15 " & VAR2
END
```

The above example will home up and clear the display. The use of row and col are relative to the screen display not memory.

RELATED FUNCTIONS

FIND
LENGTH
MID
SCREENFIND
SCREENRECT

SCREENFIND

The SCREENFIND function returns the location of a specified string. If the string begins on row 1, column 1, its location is 0,0.

SYNTAX

```
SCREENFIND (string[,startrow])
```

string
Named variable or string delimited by quotation marks.

startrow
Specifies the row in memory on which the search begins. This parameter is optional.

EXAMPLE

```
SEND SHOWME
WAITC 17
SEND VERSION
WAITC 17
SEND EXIT
WAITC 17
LET Row  = SCREENFIND("Copyright",2)
LET Row  = Row + 1
LET Var1 = "Copyright was found in row "
LET Var1 = Var1 & STRING(Row)
TELL Var1
END
```

RELATED FUNCTIONS

FIND, SCREENFIELD, SCREENRECT

SCREENRECT

The SCREENRECT function returns all the characters in the display area bounded by a specified rectangle. Variable length limit is 1000. You must restrict the length of what is returned by SCREENRECT to this length.

Note: Counting of row and column numbers is an absolute value based on terminal memory, not what is visible on the screen. Row zero may have scrolled off the top of the screen, but it is still considered row zero, and the first visible row may be some row other than zero.

SYNTAX

```
SCREENRECT (startrow,startcol[,endrow],endcol)
```

startrow
Specifies the row on which the field begins. The first row is row 0.

startcol
Specifies the column within the startrow in which the field begins. The first column is column 0.

endrow
Specifies the row in which the field ends. This parameter is optional, if the rectangle contains only one row (if startrow and endrow are equal).

endcol
Specifies the column within the endrow in which the field ends.

EXAMPLE

```
KBSPEC HP_HOMEUKEY
KBSPEC HP_CLRDKEY
SEND SHOWME
WAITC 17
SEND HELP COPY
WAITC 17
LET VAR1 = SCREENRECT(8,0,16,79)
KBSPEC HP_HOMEUKEY
KBSPEC HP_CLRDKEY
DISPLAY VAR1
END
```

In the above example, the screen is cleared. A SHOWME followed by a display of the COPY help. Using that as the display a screen rectangle based zero starting row 8 column 0 through row 16 col 79 is loaded into the variable VAR1. The display is again cleared and the selected data is displayed.

RELATED FUNCTIONS

SCREENFIND
SCREENFIELD

STRING

The STRING function changes an integer value to a string.

SYNTAX

```
STRING (integer)
```

integer
Named variable containing integer value.

EXAMPLE

```
LET Var1 = 745
LET Var2 = STRING (Var1)
LET Var2 = Var2 & " First Street"
TELL "The address is: " & Var2
END
```

In the above example, a variable named Var1 is defined with the integer 745 while a variable named Var2 uses String to change the integer to string value. The string First Street is then concatenated to the end. The content of Var2 is displayed in a dialog box with the prefix of "The address is: ".

RELATED FUNCTIONS

VALUE

SUBVER

The SUBVER function is a predefined variable that returns the current subversion number of MiniSoft 92. For example, the SUBVER of version 5.2.41 is 41.

SYNTAX

```
SUBVER
```

EXAMPLE

```
LET Var1 = STRING(MAINVER)
LET Var2 = STRING(MIDVER)
LET Var3 = STRING(SUBVER)
LET Var4 = "The current version is: " & Var1 & "." & Var2 & "." &
Var3
TELL Var4
END
```

The above example sets Var1 to the string value of the main version number, Var2 to the Mid version number, and Var 3 to the Sub version number. Var4 is set to the values of the three numbers making up the version with embedded periods, which is then displayed in a dialog box.

RELATED FUNCTIONS

MAINVER
MIDVER

UPPER

The UPPER function changes all lowercase characters in a specified string to uppercase.

SYNTAX

```
UPPER (string)
```

string

Named variable or string delimited by quotation marks. You may use the '&' operator to concatenate (join) strings.

EXAMPLE

```
LET Var1 = 'Minisoft"
LET Var2 = UPPER(Var1)
TELL Var2
    END
```

In the above example, a variable named Var1 is defined as the string "Minisoft" while a variable named Var2 uses UPPER to change the lowercase characters of the string to uppercase. The content of Var2 is displayed in a dialog box as "MINISOFT"

RELATED FUNCTIONS

LOWER

VALUE

The VALUE function changes a string containing a number to an integer value.

SYNTAX

VALUE (string)

string

Variable containing string value or string delimited by quotation marks. You may use the '&' operator to concatenate (join) strings. To convert to an integer value, the string must only contain numeric characters.

EXAMPLE

```
;*************************************************
; Convert $TIME to 12 hour format of HH:MM AM/PM
; Uses the VALUE function to convert the format
;*************************************************
LET HH = MID($TIME,1,2)
LET MM = MID($TIME,4,5)
LET TEMPHH = VALUE(HH)
IF TEMPHH > 12
  LET TEMPHH = TEMPHH - 12
  LET HH = STRING(TEMPHH)
  LET PM = "PM"
ELSE
  LET PM = "AM"
ENDIF
LET PCTIME = HH & ":" & MM & " " & PM
;*************************************************
; Set up display to show both formats
;*************************************************
LET HOLD = "$TIME in a 24hr display " & $TIME & "^M"
LET HOLD = HOLD & "Time converted to 12hr display: " &
PCTIME
TELL HOLD
END
```

RELATED FUNCTIONS

STRING

VARGET

The VARGET function updates different variables based on the argument.

SYNTAX

VARGET {COLUMN | ROW | SROW }

COLUMN as the argument updates the variable called COLUMN with the number of the current column where the cursor is located. This value is zero-relative (the first column is column zero).

ROW as the argument updates the variable called ROW with the number of the current row (where the cursor is) in display memory (as opposed to VARGET SROW, which updates with the number of the current rows on the screen). The row location returned is zero-relative (the first row is row zero).

SROW as the argument updates the variable called SROW with the number of the current row relative to the WS92 screen (as opposed to VARGET ROW, which returns the number of the current rows in display memory) The number returned is zero-relative, meaning, the first row is row zero.

EXAMPLE:

```
;*****************************************************
; Home and clear the display
;*****************************************************
KBSPEC HP_HOMEUKEY
KBSPEC HP_CLRDKEY
;*****************************************************
; Send a LISTF to get more than one page in display memory
;*****************************************************
SEND LISTF
WAITC 17
;*****************************************************
; Get the column where the cursor is located, then the Row
: where the cursor is located in Memory and Current Screen.
;*****************************************************
```

```
VARGET COLUMN
VARGET ROW
VARGET SROW
;*****************************************************
; Build the display where the location of the cursor is
; relative to one instead of relative to zero
;*****************************************************
LET V1 = COLUMN + 1
LET V2 =  "The  cursor is located in Column "
LET V2 = V2 & STRING(V1)
LET V3 = ROW + 1
LET V4 = "The cursor is located in row "
LET V4 = V4 & STRING(V3)
LET V4 = V4 & " in display memory"
LET V5 = SROW + 1
LET V6 = "The cursor is located in row "
LET V6 = V6 & STRING(V5)
LET V6 = V6 & " in current screen"
;*****************************************************
; Put the string variables together in a single display
; of three lines
;*****************************************************
TELL V2 & "^M" & V4 & "^M" & V6
END
```

The above example adds a one to all the results of the VARGET as all Row and Column values are zero-relative (the first column and row is column 0 row 0).

RELATED FUNCTIONS

SCREENFIELD
SCREENFIND
SCREENRECT

WS92 SCRIPT FILE

```
;  scrxfr.s92  7/05/2110
;
;    script to up or download a file
;    revised July 2001 for WS92 Command Language Manual
;
:domore
;--Get the name of the file on the e3000
LET HEADER ="Host file name"
LET PROMPT ="Enter the name of the file on the HPe3000"
ACCEPT          v1
;
;--Get the name of the file on the PC
LET HEADER = "Local file name"
LET PROMPT = "Enter the name of the file on the PC"
ACCEPT v2
;
;--ask which direction to go: from PC to e3000, or the opposite.
LET HEADER = "Download or Upload?"
LET PROMPT = "Enter D for download, U for upload"
ACCEPT v3
;
;--ask if this is a binary or an ASCII transfer
LET HEADER = "Binary or ASCII"
LET PROMPT = "Enter B for Binary or A for ASCII"
ACCEPT v4
IF UPPER(MID(v4,1,1)) = "A"
  ASCII
ELSE
  BINARY
ENDIF
;
;--now do the work . .
HOSTF v1
LOCF v2
IF UPPER(MID(v3,1,1)) = "D"
;--it's a download, go for it!
  DOWNLOAD
ELSE
;
;--since this is an upload, we must know the record size
```

```
   LET HEADER = "Record size"
   LET PROMPT = "Enter record size"
   ACCEPT v5
   RECSIZE v5
;
;--now do the upload
   UPLOAD
ENDIF
;
;--now ask if there's any more work . .
ASK Would you like to transfer any more files Y/N
IFYES domore
DISPLAY "All done!"
END
```

MPE/iX COMMAND FILE

```
COMMENT ---
COMMENT ---    MENU: This is a command to do a file transfer
from the
COMMENT ---          PC to the HP within menus or command
files.
COMMENT
COMMENT ---    Revised July 2001 for WS92 Command Lan-
guage Manual
COMMENT ---
SETVAR MS CHR(27)+"&oF"
SETVAR TR CHR(27)+"&oC"
SETVAR SF " "
ECHO
ECHO  ***********************************************
ECHO  *                                             *
ECHO  *   Move files between your PC and the HP3000  *
ECHO  *                                             *
ECHO  ***********************************************
ECHO
COMMENT --
COMMENT --GET UPLOAD/DOWNLOAD FROM USER
COMMENT --
SETVAR HP_RECSIZE ""
SETVAR HP_SIZE ""
INPUT TEMP_CMD; PROMPT="UPLOAD or DOWNLOAD?"
IF UPS(LFT(TEMP_CMD,1))="U" THEN
  SETVAR MS_CMD "!TR"+"UPLOAD"
  INPUT HP_RECSIZE; PROMPT="  HP record size for upload?
"
  SETVAR HP_SIZE "!MS" + "RECSIZE " + "!HP_RECSIZE"
ELSE
  SETVAR MS_CMD "!TR"+"DOWNLOAD"
ENDIF
COMMENT --
COMMENT --GET PC FILE NAME FROM USER
COMMENT --
SETVAR PC_FILE ""
ECHO
ECHO Enter PC file name (for example: A:\STUFILE.DAT)
ECHO
```

```
INPUT PC_FILE; PROMPT="  PC filespec including drive and
path? "
SETVAR PC_FILE "!MS" +"LOCF " + "!PC_FILE"
COMMENT --
COMMENT -- GET HP FILE NAME FROM USER
COMMENT --
SETVAR HP_FILE ""
ECHO
ECHO Enter HP file name (for example: STUFILE)
ECHO
INPUT HP_FILE; PROMPT="  HP filename? "
SETVAR HP_FILE "!MS" + "HOSTF " + "!HP_FILE"
ECHO
ECHO Enter type of Transfer: A for ASCII B for Binary
INPUT TYPE; PROMPT= "A or B?"
IF UPS("!TYPE") = "A" THEN
  SETVAR XFER_OPT "!MS" + "ASCII"
ELSE
  SETVAR XFER_OPT "!MS" + "BINARY"
ENDIF
COMMENT --
COMMENT --NOW EXECUTE THE COMMANDS
COMMENT --
ECHO !PC_FILE
ECHO !HP_FILE
ECHO !XFER_OPT
ECHO !HP_SIZE
ECHO !MS_CMD
COMMENT --
COMMENT --READ THE COMPLETION CODE
COMMENT --
INPUT MPE_COMMAND
IF UPS("!MPE_COMMAND") = "F" THEN
  SETVAR SF "!MPE_COMMAND"
ELSE
  !MPE_COMMAND
  INPUT SF
ENDIF
IF UPS("!SF") = "S" THEN
  IF UPS(LFT(TEMP_CMD,1))="U" THEN
    ECHO !MS TELL File Upload completed SUCCESSFULLY
  ELSE
    ECHO !MS TELL File Download completed SUCCESSFULLY
```

```
  ENDIF
ELSE
  IF UPS(LFT(TEMP_CMD,1))="U" THEN
    ECHO !MS TELL File Upload FAILED
  ELSE
    ECHO !MS TELL File Download FAILED
  ENDIF
ENDIF
```

COBOL PROGRAM

The following example initiates a file transfer calling WS92 or Session file transfer program.

```
 IDENTIFICATION DIVISION.
*
* Example HP3000 program to initiate a file
* transfer calling WS92 or Session file transfer program.
*
 PROGRAM-ID. COBXFR.
 AUTHOR. MINISOFT.
 DATE-WRITTEN. 06/26/98.
*
* revised July 2001 for command language manual.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. HP3000.
 OBJECT-COMPUTER. HP3000.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
*Variables for requesting  ID string.
*
  01 WS92-ID-REQ.
    05 FILLER                PIC X     VALUE %33.
    05 ASK                  PIC X(8)  VALUE '*s12345^'.
*
  01 WS92-ID.
    05 MAC                  PIC XX.
    05 FILLER                PIC X(19).
*
* Escape sequence to start the file transfer.
*
  01 PCFT-CMD.
    05 FILLER                PIC X      VALUE %33.
    05 FILLER                PIC XXX    VALUE '&oF'.
    05 CMD-LINE                 PIC X(90).
*
* variables to hold the Session file transfer receive string.
*
```

```
 01 SESSION-CMD.
   05 FILLER              PIC X(8)  VALUE 'RECEIVE '.
   05 FILLER              PIC X     VALUE %42.
   05 HPNAME                PIC X(26) VALUE SPACES.
   05 FILLER              PIC X     VALUE %42.
   05 FILLER              PIC X(4)  VALUE ' TO '.
   05 FILLER              PIC X     VALUE %42.
   05 PCNAME                PIC X(26) VALUE SPACES.
   05 FILLER              PIC X     VALUE %42.
   05 FILLER              PIC X(9)  VALUE ' AS TEXT '.
   05 FILLER              PIC X(7)  VALUE 'DELETE '.
*
* Variables to Move the cursor and Clear the display.
*
 01 HOME-CLR.
   05 FILLER              PIC X      VALUE %33.
   05 FILLER              PIC X      VALUE 'H'.
   05 CLR.
     10 FILLER            PIC X      VALUE %33.
     10 FILLER            PIC X      VALUE 'J'.
 01 ARROW-UP.
   05 FILLER              PIC X      VALUE %33.
   05 FILLER              PIC X      VALUE 'A'.
*
* Variable to Prompt and accept the PC and Host file names.
*
 01 HOST-FILE.
   05 FILLER              PIC X      VALUE %33.
   05 FILLER              PIC X(9)    VALUE '&oFHOSTF '.
   05 H-NAME               PIC X(26).
*
 01 PC-FILE.
   05 FILLER              PIC X      VALUE %33.
   05 FILLER              PIC X(8)    VALUE '&oFLOCF '.
   05 P-NAME               PIC X(80).
*
 01 TO-FROM.
   05 FILLER              PIC X      VALUE %33.
   05 FILLER              PIC X(3)    VALUE '&oC'.
   05 TOFROM.
     10 T-F             PIC X.
     10 FILLER            PIC X(79).
*
```

```
 01 REC-SIZE.
   05 FILLER                PIC X      VALUE %33.
   05 FILLER                PIC X(3)   VALUE '&oC'.
   05 RECORD-SIZE.
     10 FILLER              PIC X(08)  VALUE "RECSIZE ".
     10 R-S                 PIC 9(06).
     10 FILLER              PIC X(69)  VALUE SPACES.
*
 01 ASCII-BINARY.
   05 FILLER                PIC X      VALUE %33.
   05 FILLER                PIC X(3)   VALUE '&oC'.
   05 ASCIIBIN.
     10 A-B                 PIC X.
     10 FILLER              PIC X(79).
* Variables to receive and parse the RUN
* statement received from the WS92 PC to run the file transfer
* on the HP3000.
*
 77 RUN-STATEMENT           PIC X(44).
 77 DUMMY-SW                PIC XXXX.
 77 PROGRAM-NAME            PIC X(40).
 77 PARM-OPTION             PIC X(10).
 77 PARM-VALUE              PIC 999.
*
* Variable to receive the device completion code.
*
 77 DEV-COMP-CODE           PIC X.
* Variable for CREATEPROCESS intrinsic.
*
 77 CP-ERROR                PIC S9(9)  COMP.

 01 CP-ITEM-ARRAYS.
    05 ITEMNUMS-C.
       10 ITEMNUMS          PIC S9(9) COMP OCCURS 3 TIMES.
    05 ITEMS-C.
       10 ITEMS             PIC S9(9) COMP OCCURS 3 TIMES.
77  MS92LINK-PIN            PIC S9(4)  COMP.
77  SUSPEND                 PIC 9(4)   COMP.
PROCEDURE DIVISION.

MAIN-PROCEDURE.
*
*Prompt for the PC file name.
*
```

```
  MOVE SPACES TO P-NAME.
  DISPLAY HOME-CLR.
  DISPLAY "Please Enter PC's Path & Name: " WITH NO AD-
VANCING.
  ACCEPT P-NAME.
*
*Prompt for the HP File Name.
*
  MOVE SPACES TO H-NAME
  DISPLAY "Please Enter the HP file Name: " WITH NO AD-
VANCING.
  ACCEPT H-NAME.
*
*PROMPT for the Up Load or Down Load.
*
  MOVE SPACES TO TOFROM.
  DISPLAY "Please Enter If Transfer is To or From the Host: "
        WITH NO ADVANCING.
  ACCEPT TOFROM.
  IF T-F = "T" OR T-F = "t"
    THEN
      MOVE 'UPLOAD     ' TO TOFROM
    ELSE
      MOVE 'DOWNLOAD   ' TO TOFROM.
*
*PROMPT for record size if this is an upload.
*
   IF TOFROM = "UPLOAD"
      DISPLAY "RECORD SIZE for Upload"
      ACCEPT R-S
   ELSE
      MOVE ZERO TO R-S.
*
*PROMPT for the type of file (ASCII or binary).
*
  MOVE SPACES TO ASCII-BINARY.
  DISPLAY "What kind of file: ASCII or binary (A or B)".
  ACCEPT ASCIIBIN.
  IF A-B = "A" OR "a"
     MOVE "ASCII" TO ASCIIBIN
  ELSE MOVE "BINARY" TO ASCIIBIN.
*
*Request, receive, and check for WS92 ID string.
```

```
*
 MOVE SPACES TO WS92-ID.
 DISPLAY WS92-ID-REQ.
 ACCEPT WS92-ID.
 IF WS92-ID IS EQUAL TO "MS92 BEST"
   THEN
    PERFORM WS92XFER
    PERFORM RUN-LINKPROG
    PERFORM CLEANUP
    STOP RUN.
 IF WS92-ID IS EQUAL TO "70092"
   THEN
    MOVE "*s811^  " TO ASK
    MOVE SPACES TO WS92-ID
    DISPLAY WS92-ID-REQ
    ACCEPT WS92-ID
    IF MAC IS EQUAL TO "AM"
    THEN
     PERFORM SETUP-SESSION
     PERFORM RUN-LINKPROG
     PERFORM CLEANUP
     STOP RUN.
* Must be WRQ or HP Terminal
 DISPLAY "Emulator is not WS92 or Session".
 STOP RUN.
CLEANUP.
 IF DEV-COMP-CODE IS NOT EQUAL TO "S"
   THEN
    DISPLAY "File Transfer did not Complete"
    STOP RUN.

*
* End of Program.
*
   DISPLAY ARROW-UP WITH NO ADVANCING.
   DISPLAY ARROW-UP WITH NO ADVANCING.
   DISPLAY ARROW-UP WITH NO ADVANCING.
   DISPLAY ARROW-UP WITH NO ADVANCING.
   DISPLAY ARROW-UP WITH NO ADVANCING.
   DISPLAY ARROW-UP WITH NO ADVANCING.
   DISPLAY CLR.
   DISPLAY "File Transfer is completed.".
*
```

```
 WS92XFER.
*
* Send the File Transfer command to the WS92.
*
   DISPLAY PC-FILE.
*
   DISPLAY HOST-FILE.
*
   MOVE SPACES              TO CMD-LINE.
*
   DISPLAY TO-FROM.
   MOVE SPACES              TO CMD-LINE.
   IF R-S NOT EQUAL ZERO
     DISPLAY REC-SIZE.
*   MOVE "ASCII"             TO CMD-LINE.
*   DISPLAY PCFT-CMD.
   DISPLAY ASCII-BINARY.
   ACCEPT RUN-STATEMENT.
* Session for Macintosh file transfer.
 SETUP-SESSION.
  MOVE P-NAME TO PCNAME.
  MOVE H-NAME TO HPNAME.
  MOVE SESSION-CMD TO CMD-LINE.
  DISPLAY PCFT-CMD.
  ACCEPT RUN-STATEMENT.
 RUN-LINKPROG.
   MOVE SPACES              TO PROGRAM-NAME,PARM-
OPTION.
   MOVE ZEROS               TO PARM-VALUE.
   UNSTRING RUN-STATEMENT
     DELIMITED BY ALL SPACE, OR ";"
     INTO DUMMY-SW, PROGRAM-NAME, PARM-OPTION.

   IF PARM-OPTION IS NOT = SPACES
     UNSTRING PARM-OPTION DELIMITED BY ALL SPACE,
OR "="
      INTO DUMMY-SW, PARM-VALUE.

   MOVE 2                   TO ITEMNUMS (1).
   MOVE PARM-VALUE          TO ITEMS(1).
   MOVE 3                   TO ITEMNUMS(2).
   MOVE 1                   TO ITEMS(2).
   MOVE 0                   TO ITEMNUMS(3).
```

```
MOVE 0                  TO ITEMS(3).
MOVE 2                  TO SUSPEND.
CALL INTRINSIC "CREATEPROCESS"
    USING CP-ERROR, MS92LINK-PIN,@PROGRAM-NAME,
        ITEMNUMS-C, ITEMS-C.

CALL INTRINSIC "ACTIVATE"
     USING \MS92LINK-PIN\, \SUSPEND\.
ACCEPT DEV-COMP-CODE.
```