MicroPhone II for Windows

Help Index

The Index contains a list of all Help topics categories available for MicroPhone II. Index items are listed within each major category.

For information on how to use Help, press F1 or choose Using Help from the Help menu.

Command Menus

MicroPhone II's eight command menus contain all the commands and options that operate the program. You may view one menu at a time; each menu will remain on the screen until you choose a command or actively retract the menu.

Keyboard

Script Language

DDE

Dynamic Data Exchange (DDE) is a message-passing tool provided by the Windows operating system. Windows applications use DDE to exchange data in real time.

The simplest way to establish a DDE link with MicroPhone II is to use the Copy Link and Paste Link functions in the Edit Menu.

MicroPhone II also provides full support for DDE conversations with its script language. The DDE * ... script command is provided to send commands or information to other applications. With MicroPhone II and DDE, you can automate MicroPhone II, as well as any application that supports DDE.

See the following topics for more information:

DDE Messages
Starting Another Application from MicroPhone II
Responding to Another Application
Getting Data From Another Application
Executing Commands in Another Application
Using Quotation Marks

Screen Regions

Title Bar
Maximize Box
Minimize Box
Control-enu
Menu Bar
Terminal Window
Scroll Bars
Button Bar
Abort Buttons
Connect Button
Pause Button

File Menu

Commands in the File Menu are used to create, open and save MicroPhone II settings documents; to direct selected text to a disk or to the printer, either during a session or after a session is completed; and to quit MicroPhone II.

For more information, select the File menu command name.

New Settings	Opens a new, untitled settings document.
Open Settings	Opens an existing settings document.
Close Settings	Closes the current settings.
Save Settings	Saves the current settings to an existing file.
Save Settings as	Saves the current settings to a new file.
Open Capture File	Opens a file for saving text from the terminal window.
Append to Capture File	Adds text from the terminal window to an existing file.
Stop/Resume Capturing to	Stops or resumes data capture.
Save Selection As	Saves selected text from the screen to a text file.
Append Selection	Adds selected text from the screen to an existing file.
Start Printing/Stop Printing	Starts or stops sending text from the screen to the printer.
Print Selection	Prints the text selected on the screen.
Clear Print Buffer	Prints data in the Print Buffer to the printer.
Setup Printer	Sets up the printer through the page setup dialog box.
Exit	Ends a session with MicroPhone II.

Edit Menu

Edit Menu commands allow you to copy and paste data from the screen, as well as select and clear data from the screen buffer.

For more information, select the Edit menu command name.

Cut	Deletes selected text and moves it to the clipboard.
Copy	Copies selected text to the clipboard.
Paste	Moves text from the clipboard to the terminal window.
Delete	Deletes selected text without moving it to the clipboard.
Copy Link	Copies data from MicroPhone to an application linked via DDE.
Paste Link	Paste data to MicroPhone from an application linked via DDE.
Copy Table	Copies a table to the clipboard.
Select All	Selects all the text on the screen.
Clear Buffer	Clears all information from MicroPhone II's screen buffer.

Settings Menu

MicroPhone II's Communications, Terminal, Text Transfer, Protocol Transfer, Startup Action and System Preferences settings allow you to customize the program for your particular telecommunications needs.

For more information, select the Settings menu command name.

Communications	Opens dialog box for setting your modem settings.
Terminal	Opens dialog box for setting your terminal parameters.
Text Transfer	Opens dialog box for setting your text transfer protocols.
Protocol Transfer	Opens dialog box for setting your file transfer protocols.
Startup Action	Opens dialog box for startup action settings.
Restore Settings	Restore all settings to their original values
System Preferences	Opens dialog box for settings that apply to general MicroPhone II operations.
Character Sets	Opens dialog box for selecting a character set.
Reset Comm Facility	Re-initializes the port.

Phone Menu

The Phone commands allow you to create and modify your phone directory. The modem commands include Dial, Redial, Wait for Call and Hang Up. In addition, the Phone menu allows you to attach a script of your choice to a specific phone number so that the script is activated as soon as a connection is established.

For more information, select the Phone menu command name.

Create Service	Leads to a dialog box for creating a dial service.
Modify Service	Leads to a dialog for selecting a service to be modified.
Delete Service	Leads to a dialog for selecting a service to be deleted.
Connection Open/Close	Opens/Closes the current connection.
Wait for Call	Set up modem for an incoming call.
Hang Up	Instructs the modem to hang up.
Redial	Dial the service continuously until a connection is established.
Dial Service	Dial the service created for a specific settings document.

Scripts Menu

Commands within this menu allow you to create scripts using either Watch Me (the automatic recording mode) or MicroPhone II's Script Editor. MicroPhone II also allows you to create a script using your favorite text editor and then import it into a settings document of your choice.
(If you are a user of MicroPhone on the Macintosh, you may want to share scripts between machines. See Appendix in Reference Manual on Macintosh to Windows.)

For more information, select the Scripts menu command name.

Watch Me	Record a session in a script.
Edit Script	Brings up the Script Manager Dialog.
Trace	Use this command to find errors in scripts.
Do Script	Select a script for execution.

Transfer Menu

This menu allows you to send and receive any sort of file: text, graphics, spreadsheets, page layout, databases and software. You may also specify the source and destination directories of the files to be transferred.

For more information, select the Transfer menu command name.

Send	Leads to a directory dialog box for selecting a file to send and a protocol to use.
Receive	Leads to a directory dialog box for selecting a directory in which to save and a protocol to use.
Batch	Invokes the batch editor to create a group of files for batch transfers.
Select Receive Directory	Name the default directory to receive files.

Window Menu

When more than one MicroPhone II session is open, the Window menu appears. It contains a list of all the open MicroPhone II sessions. Choosing a particular session from this menu will bring the focus to that session.

Help Menu

MicroPhone II's Online Help.

Index	MicroPhone II Help Index.
Commands	MicroPhone II Menu Commands.
Keyboard	MicroPhone II Keyboard Usage.
Script Language	MicroPhone II Script Language.
DDE	MicroPhone II DDE Usage.
Screen Regions	MicroPhone II Screen Regions.
About MicroPhone II	Displays the program's name, authors and version.

Script Commands

Trace
Until
Wait
Wait for Call
When
While

Script Expressions

An expression may be a single constant or variable name, or a combination of constants, variables, operators and other elements. These are some typical expressions:

```
(102 + b) * 5
POS('xyz', ST)
J >= 7
'abc' & bg & 'xyz'
```

See also: Elements of an Expression
Expression Syntax
Variables
Operators
Functions

Script Operators

Operator Precedence
AND
DIV
MOD
NOT
OR
XOR
* (Multiplication)
/ (Division)
- (Subtraction)
+ (Addition)
= > < <= >= <> (Comparison Operators)
& (Ampersand)

Script Functions

TickCount
TypeOf(e)
UCase(s)
Year

Logical Constructs

A logical construct is a group of script commands that change the flow of script execution. The group contains at least one decision command to check a specified condition. If the condition is true, the script continues along a particular path. If the condition is false, the script continues along a different path.

MicroPhone II's four logical constructs - If, Repeat, When and While - follow structured programming conventions as found in Pascal and Modula 2. Note the following:

Each of the constructs may be used whenever a single script command may be used.

There is no specific limit to the number of statements within each construct.

Wait and the When construct are "dynamic." They repeatedly test the conditions that they define. After each test or set of tests, they let a character move from the input port to the screen. Use these commands to look for specific strings in the incoming data.

The constructs If, While, and Repeat are "static." They test the conditions they define but do not move characters to the screen. Use them only after data transmission is paused or has ceased.

Only three of MicroPhone II's script commands - Wait, When and Send File * - "open the gate," so to speak, so that data can flow into your terminal window. Wait looks for a single specified condition, and when it occurs, stops incoming data. The When command does the same for a number of specified conditions. As soon as one is encountered, it stops incoming data. Finally, Send File * sends characters out and allows echoed characters coming in to appear on the screen at the same time.

All four logical constructs use the **text equals** and **line contains** tests. **Text equals** tests for equivalence on the current line (for example, when a prompt expects your answer on the same line). **Line contains** tests for equivalence on the previous line, as when a prompt is followed by a carriage return, which expects your answer on a new line.

A **line contains** test will return False if the cursor is not positioned in column one.

Success Flag

Script commands noted with an asterisk (*) set a Success flag to either True or False (except where noted) depending on whether or not the action they initiate has been completed successfully. This flag is actually an internal variable (called "Success") of boolean value (True or False).

The Success flag may be tested by any of MicroPhone II's logical constructs . If you want to test the Success flag, do it directly after the command that sets it. Do not assume that the flag remains unchanged between commands. (Some MicroPhone II functions also set Success.)

File New Settings Command

This command opens a new, untitled settings document that contains MicroPhone II's default settings.

The default values for each setting are stated in **The Settings Menu** chapter in the Reference Manual.

See also: Open Settings
Close Settings
Save Settings
Save Settings as
Open Capture File

File Open Settings Command

This command presents a dialog box which lists all the settings documents that have been installed.

Select the name of a document and then choose **OK** to open the document.
The **Close current settings** option determines whether the current settings document will be closed. If it is not selected, the new settings document is opened in another MicroPhone II window. Deselecting this option allows you to run more than one MicroPhone II settings document at a time.

If the **Close current settings** option is selected when you choose the **OK** button, the current settings document is closed. This is the same as choosing **Close Settings** from the File menu before choosing **Open Settings**.

See also: New Settings
Close Settings
Save Settings
Save Settings as
Open Capture File

File Choose Command

This dummies the File Choose command.

File Close Command

This command closes the active settings document. If you have changed any settings, the program will ask if you want to save them.

Choose **Yes** to save the document with all changes to services, scripts and settings that you made during the session. Text which is on the screen or in the screen buffer, however, is not part of a settings document, and will be lost unless printed or saved to a capture file.

Choose **No** to close the document without saving any changes. **Cancel** returns you to the active document without saving it.

See also: New Settings
Open Settings
Save Settings
Save Settings as
Open Capture File

File Save Settings Command

The **Save Settings** and Save Settings as... commands will save the current settings to disk. Use the **Save Settings** command to update a document that is already stored on disk. Use Save Settings as... for a new document that must be named for the first time, or to keep the original settings document unchanged and save a new version with the recent changes to another file.

See also:
New Settings
Open Settings
Close Settings
Save Settings as
Open Capture File

File Save Settings as Command

The Save Settings and **Save Settings as...** commands will save the current settings to disk. Use the Save Settings command to update a document that is already stored on disk. Use **Save Settings as...** for a new document that must be named for the first time, or to keep the original settings document unchanged and save a new version with the recent changes to another file.

Choosing the **Save Settings as...** command leads to a dialog box where you can enter in the name under which you can save the doucment.To save the document, type the new name in the **Name** text box and choose **OK**. MicroPhone II will automatically generate a unique filename to identify the file holding the document.

The **All Names** list box shows the names of all the settings documents currently installed in MicroPhone II. Clicking a name in this list will place it and its associated filename in the Name and File text boxes.

To change the name of the current settings document, click its name in the **All Names** list box. This will enter the filename in the **File** text box. Then enter the new name in the **Name** text box and choose **OK**. MicroPhone II will warn you that you are about to overwrite an existing file and give you a chance to back up. If you don't cancel, the new name will replace your old one.

Choose the **File** button to change the location on your disk for storing the file containing the named settings document. A directory dialog box is presented allowing you to specify the directory.

Type the filename in the **File** text box. The name may be eight characters with a three-character extension after the period. If you do not include the period, the default extension shown in the path will be appended. (If you use the "**.MDC**" extension, your document will automatically appear in the Files list box whenever you open a settings document.)

You may also enter a path as part of your document's filename in order to save it to another drive or directory. The text box allows up to 64 characters.

See also: New Settings
Open Settings
Close Settings
Save Settings
Open Capture File

File Open Capture File Command

This command opens a capture file for saving the text that appears in the terminal window. You may open a new file, or overwrite an existing one. This command leads to a directory dialog box similar to that presented by the Save Settings as... File command

The **OK** button accepts the filename in the **File** text box and returns to MicroPhone II's terminal window. From this point on, all information added to the session, both incoming and outgoing, will be saved to disk in this file until you choose the Stop Capturing to command.

If you are saving data to an existing file, you will be presented with a message asking whether you want to overwrite the existing file. Choose the **OK** button to replace the old file with the new one. Choose the **Cancel** button to return to the Open Capture File dialog box and type another name.

If a capture file is already open, this command automatically closes it and saves all the information that was captured.

Captured text is saved in ASCII text files. The files may be accessed by most word processing programs or any text editor, such as Windows' Notepad.

See also: Append to Capture File
Stop /Resume Capturing to

File Append to Capture File Command

This command opens a capture file for saving the text that appears in the terminal window. You may open a new file, or overwrite an existing one. This command leads to a directory dialog box

The **OK** button accepts the filename in the **File** text box and returns to MicroPhone II's terminal window. From this point on, all information added to the session, both incoming and outgoing, will be saved to disk in this file until you choose the Stop Capturing to command.

If you are saving data to an existing file, you will be presented with a message asking whether you want to overwrite the existing file. Choose the **OK** button to replace the old file with the new one. Choose the **Cancel** button to return to the Open Capture File dialog box and type another name.

If a capture file is already open, this command automatically closes it and saves all the information that was captured.Captured text is saved in ASCII text files. The programs or any text editor, such as Windows' Notepad.

See also: Open Capture File
Stop /Resume Capturing to

File Stop Capturing to/Resume Capturing to Command

The commands Open Capture File... and Append to Capture File... automatically activate text capture. To stop text capture, choose **Stop Capturing to** from the File menu. This closes the capture file, making it available to other programs. Then the menu command will change to read **Resume Capturing to**, and you can choose this to restart text capture into the same file. The command will toggle between stop and resume until you close the session. In this way, you can save selected portions of a session.

See also: Open Capture File
Append to Capture File

File Save Selection as Command

This command will save selected text from the screen to a text file. In the File menu, this command is greyed until text is selected. Once a portion of text has been selected, you can choose **Save Selection as...**. The command leads to a dialog box similar to that presented by Open Capture File....

Specify a filename and choose the **OK** button to save your selection. Choose the **Cancel** button to return to the terminal window without saving the selection.

Selections are saved as ASCII text files. They may be accessed by most word processing programs or any text editor, such as Windows' Notepad.

File Append Selection Command

This command is similar to the Save Selection as... command. **Append Selection to...** adds selected text to an existing text file instead of creating a new one. This command is greyed until text is selected. The command presents a directory dialog box

To open a selection text file, first choose the name of the file, and then click the button labeled **OK**. The selected text will be added to the end of the chosen file.

File Start Printing/Stop Printing Command

When **Start Printing** is chosen, any information scrolling into the terminal window is sent to your printer. An Abort Print button will appear at the bottom of the work area, to the right of the Pause button. To stop printing, choose the abort button, or pull down the File menu and choose the command that now reads **Stop Printing**.

The printout will be in the printer's text mode. If you want to format the data, first use one of MicroPhone II's save-to-disk features (Open Capture File..., Append to Capture File... or Save Selection as...) to store the information in a text file, and later open the file with a word processing program.

IMPORTANT:
Text sent to the printer is first loaded into the print spooler's buffer. The buffer holds the text until the printer is ready for it. Printing will not begin until you choose **Stop Printing** or until this buffer is full.

Be sure the printer is connected and on-line before you choose the **Start Printing** command. Otherwise, when the spooler's buffer is full, an error message will appear.

If you see the error message, ensure the printer is ready and then choose the **Retry** button. If you have to choose **Cancel**, the spooler will close and the information in its buffer will be lost. Then, in order to print, you will have to scroll back into the terminal window's buffer, select the text for printing and use the Print Selection command -- after your printer is connected and on-line.

File Print Selection Command

The **Print Selection** command will print the text selected on the screen. In the File menu, this command is greyed until data is selected. Once text has been selected, **Print Selection** may be chosen.

The printer must be connected and on-line. If you want to cancel the print job, use the Abort Print button or choose the cancel button in the "Printing to Spooler" alert box.

The printout will be in the printer's text mode. If you want to format the data, first use one of MicroPhone II's save-to-disk features (Open Capture File..., Append to Capture File... or Save Selection as...) to store the information in a text file, and later open the file with a word processing program.

File Clear Print Buffer Command

This command completes a print job in VT102 mode. When a Print Line command is issued from the remote terminal and you are not running a script, choose this command to "flush" out the last line of data to the printer. As a menu item, **Clear Print Buffer** is greyed until the VT102 Print Line command is received.

File Setup Printer Command

This command allows you to access the page setup dialog box for the printer that is currently chosen. (See your Windows documentation for more information on how to choose another printer.)

File Cancel Command

This dummies the File Cancel command.

File Reply Command

This dummies the File Reply command.

File Memory Command

This dummies the File Memory command.

File Monitor Command

This dummies the File Monitor command.

File Exit Command

The **Exit** command ends all MicroPhone II sessions. If changes were made to any of the services, scripts or settings in any open document, and these have not yet been saved, the program will present the following message for each document that has changed:

> "Current settings have not been saved. Do you want to save them to '(document name).mdc'? "

If you choose **Yes**, the program saves the document to reflect changes made to services, scripts and settings since the last time the document was saved. Text on the screen or in the screen buffer, however, is not part of a settings document, and will be lost unless printed or saved to a capture file. (Choose the **Cancel** button to return to the terminal window to save text.)

If your document is untitled, you will be asked to name it. The MicroPhone II program will then close.

The **No** button closes the settings document without changing any of the document's settings.

The **Cancel** button returns you to the current session without exiting the program or saving any changes in the documents still open. Settings documents that have already been closed remain closed.

File Stack Command

This dummies the File Stack command.

File Timer Command

This dummies the File Timer command.

Edit Editor Command

The commands in this menu use the **Clipboard** to copy and paste text in the work area of your terminal window and screen buffer. **Notepad**, as discussed in your Windows documentation, is a good place to examine and prepare text. Using **Windows**' **Clipboard**, you can cut or copy information from one application and then paste it into the **Notepad** or directly into your MicroPhone II session.

Edit Cut Command

The **Cut** command deletes selected text and moves it to the Clipboard. This replaces any data that might already be on the Clipboard as the result of a previous action to the Clipboard.

If the cut is performed in the terminal window, text is simply removed and replaced with white space. If **Cut** is invoked in the screen buffer, text is removed and remaining text closes up, eliminating the white space. This is handy for cleaning up text before saving the buffer's contents to a file.

Edit Copy Command

The **Copy** command duplicates selected text and places the copy on the Clipboard. The duplicate replaces any data that might already be on the Clipboard as the result of a previous action to the Clipboard.

Edit Paste Command

Pasting text from the Clipboard to the terminal window is like typing or sending text (ASCII) in the window yourself. Text is pasted to the left of the cursor, and, if you are on-line, it will be sent to the remote computer.

Edit Delete Command

The **Delete** command deletes selected text. Unlike the Cut command, the deleted text is not moved to the Clipboard.

As with the Cut command, if text is cleared in the terminal window, it will be replaced by white space. If cleared in the screen buffer, remaining text will close up and eliminate white space.

Edit Copy Link Command

The **Copy Link** command allows you to create a DDE "hot link" from a MicroPhone II variable to another Windows application. When a copy link is established, any changes to the variable within MicroPhone II will automatically update the copy in the linked application. This is used to continually update other applications with the contents of variables as they are changed by MicroPhone II.

Choosing this command displays a list of all the variables in MicroPhone II that may be linked to another application.

These variables include a number of MicroPhone II's settings functions, as well as any variables created by MicroPhone II scripts. Additionally, **Input** sends a copy of data from the remote computer to the DDE application, and **Output** sends a copy of MicroPhone II's outgoing text to the DDE application.

When you choose one of these variables, the link information is copied to the Clipboard. This will replace any other information that may have already been on the Clipboard. You may then open the other application and choose Paste Link from its Edit menu to establish the link. MicroPhone II will sever the link when either of the applications are exited or the MicroPhone II variable is deleted.

MicroPhone II's script commands provide more control over the use of DDE. **Copy Link** is very similar to MicroPhone II's DDE Advise script command.

Edit Paste Link Command

The **Paste Link** command allows you to create a DDE "hot link" from another Windows application to a MicroPhone II variable. When a paste link is established, any changes to the item within the linked application will automatically update the copy in the MicroPhone II variable. This is used to continually update a MicroPhone II variable with the contents of the linked item as it is changed by the linked application. The command is greyed until link information has been copied to the Clipboard.

Choosing this command displays a list of all the variables in MicroPhone II that may receive a link from another application. These include a number of MicroPhone II's settings functions, as well as any variables created by MicroPhone II scripts. Additionally, **Input** inserts data from the DDE application into the terminal window. The data is interpreted as if it came from a remote computer. **Output** sends the data from the DDE application to the remote computer as if it had been typed into the terminal window.

When you choose one of these variables, the link information is copied from the Clipboard. The link remains open until either of the applications are exited, the MicroPhone II variable is deleted or the link information is deleted from the other application.

MicroPhone II's script commands provide more control over the use of DDE. **Paste Link** is very similar to the DDE Advise command.

Edit Copy Table Command

This command is useful for transferring tabular information to spreadsheet and data base programs that expect tab delimited columns. The **Copy Table** command copies a table to the Clipboard, replacing each occurrence of two or more consecutive space characters with a tab character. To copy a table into another application, follow these steps:

Select the tabular data.

Choose the **Copy Table** command. The data is copied to the Clipboard with each occurrence of two or more spaces replaced with a single tab character.

Launch the other application and paste the table as desired.

Edit Select All Command

This command selects all the text in the screen buffer and in the terminal window. This is a handy method for selecting all the text before a Save Selection As... or Print Selection command.

Edit Clear Buffer Command

This command clears all information from MicroPhone II's screen buffer. Before data is cleared, you'll be prompted with a warning to avoid inadvertent loss of data. **Clear Buffer** does not delete the text in the terminal window.

For more information about the screen buffer, see System Preferences...

Edit Test Command

This dummies the Edit Test command.

Edit Clear Command

This dummies the Edit Test command.

Window Command

This command brings you to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window Command

This command brings you to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window Command

This command brings you to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window Command

This command brings you to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window Command

This command brings you to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window Command

This command brings you to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window Command

This command brings you to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window Command

This command brings you to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window Command

This command brings you to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window Command

This command brings you to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window More Command

This command brings you to a dialog box where you can choose to go to a currently opened MicroPhone II Settings Document.

See Window Menu.

Window Win1 Command

This dummies the Window Win1 command.

Window Win2 Command

This dummies the Window Win2 command.

Settings Communications Command

This command leads to a dialog box where you may change options for setting your modem to work with a particular service, or for preparing for a direct connection. The communications parameters are:

Modem
Connection
Baud Rate
Bits per Character
Parity
Stop Bits
Flow Control

Install Modem Driver
The Install Modem Driver button provides access to the files that contain new modem drivers not included in the original MicroPhone II package.

Install Connection Driver
The Install Connection Driver button provides access to files that contain new connection drivers not included in the original MicroPhone II package. (See the Editing Communications Drivers chapter in the User's Guide for more information.)

For more information on each parameter, see the topic on Communications Settings Dialog Box.

Settings Pick Terminal Command

This dummies the Settings Pick Terminal command.

Settings Terminal Command

This command lets you set the parameters for you terminal. The parameters are:

Terminal Type
Columns
Rows
Font
Size
Cursor Shape
Backspace
Answerback
Capture on CR
Capture on Clear
Strip 8th bit
Local Echo
New Line
Auto Wraparound

For more information on each parameter, see the topic on Terminal Settings Dialog Box.

Settings Terminal Special Command

This dummies the Settings Terminal Special command.

Settings Text Transfer Command

This command leads to a dialog box where you may change the settings for MicroPhone II's text transfer protocols. The file transfer is initiated from the Transfer menu. Default values of these settings should apply to most transfers. The settings parameters are:

Flow Control:
- X-on/X-off Pacing
- Wait for Echo
- Wait for Prompt Char
- Delay between Chars
- Delay between Lines

Word-wrap Outgoing Text Column
Line terminator for Outgoing Lines
Line terminator for Captured Lines

For more information on each parameter, see the topic on Text Transfer Settings Dialog Box.

Settings Protocol Transfer Command

This command leads to a dialog box where you may change the settings for MicroPhone II's file transfer protocols. The file transfer is initiated from the Transfer menu. Default values of these settings should apply to most transfers. The Protocol Transfer Settings are:

Text
XMODEM
YMODEM
YMODEM-G
ZMODEM

For more information on each parameter, see the topic on Protocol Transfer Settings Dialog Box.

Settings Startup Action Command

This command allows you to set MicroPhone II to automatically dial a service or run a script each time this document is opened.

When the appropriate settings document is open, choose **Startup Action...**. A dialog box will appear (it will be empty if you don't have any scripts or services)

The **None**, **Dial Service** and **Do Script** option buttons control what startup action, if any, is initiated. Only one action may be set at a time.

None is the default setting and, if selected, will not initiate any automatic action.

Under **Dial Service**, you will see a list box with the service names,if any, defined for the document. You may select one.

Under **Do Script**, you will see a list box with all the script names defined for the document. You may select one.

See also: Startup Action Dialog Box.

Settings Restore Settings Command

This command will restore all settings to their original values, reversing any unsaved changes made during the current session.

Settings System Preferences Command

This command opens dialog box where you might change settings that apply to general MicroPhone II operations. The settings are:

Size of Screen Buffer
Display Buffer Full (Yes/No)
Default Document Name
Keep Connection Open (Yes/No)

For more information on each parameter, see the topic on System Preferences Dialog Box.

Settings Character Sets Command

This commands opens the dialog box for selecting a character set. The current character set is selected. To choose another character set, simply select the new one and click OK. The default is the ANSI set.

This command does not translate your English text into another language! It simply replaces a standard character with the international character that is mapped to that ASCII code.

When incoming text is saved to a file, the characters are mapped to the international characters available in MicroPhone II's ASCII font. When you open that text file with a word processing program, the international characters are preserved. When sending text, MicroPhone II does the opposite; it takes your international characters and maps them to the correct international character code for transmission.

For more information on each parameter, see the topic on Character Set Translation Dialog Box.

Settings Reset Comm Facility Command

When the port has been told to stop sending data (X-off), and there is no subsequent command to resume sending data (X-on), the user cannot communicate in either direction. To re-initialize the port, issue the command Reset Comm Facility; this will clear the communications port, and data transmission will resume.

Phone Create Service Command

This command allows you to enter a new name and phone number in your "telephone directory" (settings document). Each service is composed of a name, a phone number and a dialing mode. At your option, you may designate any of your scripts to be automatically invoked once the connection is established. There is no specific limit to the number of entries a single document may contain.

To create a new service, follow these steps:

In the **Service Name** box, enter the name - up to 24 characters - you want to assign to this phone number.

In the **Phone Number** box, enter the phone number - up to 255 characters - as you would dial it.

A phone number may be entered in any of the following formats: 4158491912, (415) 849-1912 or 415-849-1912.

On most modems, entering a comma (,) causes the modem to pause two seconds. If, for instance, you enter the number 9,849-1912, the modem will dial 9, then pause for two seconds, and dial the rest of the number.

If you enter the at-symbol (@) followed by a filename instead of a phone number, MicroPhone II will prompt you for the phone number when you first call this service. The number will then be stored in the named file for use on subsequent calls. Later, if you want to change the number, you must delete the file and MicroPhone II will prompt you for a new phone number.

Select the **Dialing Mode**.

The choices of **Tone**, **Pulse** and **Mixed** allow you to use MicroPhone II on any telephone line in the U.S. and in most parts of the world. Use Pulse if your phone does not support touch-tone. Touch-Tone is the default. Choose Mixed if your residential line does not have touch-tone but the central switching office does.

When **Mixed** is selected, you should indicate which digits are to be dialed in which mode by typing a "P" for Pulse and a "T" for Tone before the respective digits in the Phone Number box.

If you wish to invoke a specific script automatically after you dial this phone number, check the box next to **After connecting, do script**. Then select your script from the list box. This box lists all the scripts in the open settings document.

IMPORTANT:
Don't use the Dial Service script command in the script. If you do, the program will first dial the service, then activate the script and attempt to dial the same service again. That will put the program in a never ending loop.

Service names will be displayed under Dial Service. To dial a number, pull down the Phone menu and select the service you want to call as you would select any other menu command.

See also: Modify Service
Delete Service

Phone Modify Service Command

Leads to a dialog for selecting a service to be modified. Once you have selected a service, you will be presented with a dialog box similar to the one used to create the service. Follow the steps outlined in Create Service command to modify the service.

Phone Delete Service Command

Leads to a dialog for selecting a service to be deleted.

See also: Create Service
Modify Service

Phone Dialing Directory Command

This dummies the Dialing Directory Menu -- a new feature.

Phone Connection Open Command

Use this command to manually open a connection to the connection port chosen under Communications Settings. You can then type directly to your modem or whatever is connected to the serial port. This is the same as choosing the Connect button to the left of the horizontal scroll bar. The Connect button changes to Pause and the Phone menu Connection Open changes to Connection Close. To close the connection, choose Connection Close

MicroPhone II will automatically open the connection when you choose a service to dial. Also, MicroPhone II will normally release the connection port for others to use when you choose Hang Up. This allows other applications or network users to share the same connection. However, the automatic close can be disabled under System Preferences... in the **Settings Menu**.

IMPORTANT:
When the connection is open, no other application can use the connection port because MicroPhone II is using it. If you are using a network modem, it cannot be used by anyone else. MicroPhone II will only close the connection port when you close the settings document.

Phone Connection Close Command

Use this command to manually close a connection to the connection port.

See also: Connection Open.

Phone Wait for Call Command

Use this command to prepare your modem for an incoming call. **Wait For Call** sets up the modem so that it’s ready to "pick up the phone." Then you can leave your computer unattended while waiting for a connection to be made.

Phone Hang Up Command

This command instructs the modem to hang up (by sending the "ATH" code) and disconnects you from the remote computer.

IMPORTANT:
If you can, don’t hang up without first issuing the proper log-off command when one is required. If you do not follow the proper procedure on a network, the time-sharing computer may not notice your absence and will continue to charge you for some time after you have "hung up."

Phone Redial Command

When a Dial Service command is unsuccessful, or the dialing procedure is interrupted or incomplete, the service name will appear next to the Redial command in the Phone menu. Choose **Redial** to attempt to connect again. This command will dial the service continuously until a connection is established.

In the Phone menu, Redial is greyed until the first time that a Dial Service command is attempted.

Phone Dial Service Command

Service names created for a specific settings document will be displayed under the grey **Dial Service:** heading in alphabetical order. To dial, pull down the Phone menu and select the service name.

However, if you have more service names than will fit into the Phone menu, this command will change to read **Dial Service...** and is no longer grey.

Choosing **Dial Service...** leads to a list box that displays service names alphabetically. Selecting a name will initiate the **Dial Service** command.

See also: Redial

Phone Service Command

This command will cause MicroPhone II to dial the service that you have selected.

See also: Phone Dial Service
Phone Create Service

Phone Service Command

This command will cause MicroPhone II to dial the service that you have selected.

See also: Phone Dial Service
Phone Create Service

Phone Service Command

This command will cause MicroPhone II to dial the service that you have selected.

See also: Phone Dial Service
Phone Create Service

Phone Service Command

This command will cause MicroPhone II to dial the service that you have selected.

See also: Phone Dial Service
Phone Create Service

Phone Service Command

This command will cause MicroPhone II to dial the service that you have selected.

See also: Phone Dial Service
Phone Create Service

Phone Service Command

This command will cause MicroPhone II to dial the service that you have selected.

See also: Phone Dial Service
Phone Create Service

Phone Service Command

This command will cause MicroPhone II to dial the service that you have selected.

See also: Phone Dial Service
Phone Create Service

Phone Service Command

This command will cause MicroPhone II to dial the service that you have selected.

See also: Phone Dial Service
Phone Create Service

Phone Service Command

This command will cause MicroPhone II to dial the service that you have selected.

See also: Phone Dial Service
Phone Create Service

Phone Service Command

This command will cause MicroPhone II to dial the service that you have selected.

See also: Phone Dial Service
Phone Create Service

Phone Modem Command

MicroPhone II has five modem commands: Connection Open/Close, Wait for Call, Hang Up, Redial, and Dial Service.

They are actually built-in scripts that tell MicroPhone II how to communicate with the modem. These scripts employ the AT Command Set (described in Modem Conventions Appendix). If your modem does not respond to standard AT commands, you can override the internal modem scripts by creating your own modem driver. (Refer to the Editing Drivers chapter in the User's Guide for more information.) These commands are also accessible from scripts.

Phone Connect Command

See Connection Open, and Connection Close.

Scripts Watch Me Command

The **Watch Me** Command brings up a dialog box for creation of a script by watching your key strokes and commands you are going to use in the terminal window. You can see a script being generated in the Watch Me editor while you conduct the session. You may also edit the script that is still in the Watch Me editor. To save the new script, return to the **Watch Me** window and choose the **Close** button. To abort the **Watch Me** session, return to the **Watch Me** window and choose **Cancel**.

Scripts Edit Script Command

This command opens the Script Manager, where all the scripts in your settings document are displayed. The Script Manager allows you to perform a variety of tasks related to scripting, including create, modify, copy, import and export.

Scripts Trace Command

Use this command to find errors in scripts. Trace is activated by choosing **Trace...** from the menu before you execute a script. This will bring up a dialog box where you can set the trace options:

Trace Execution	To use Trace, check the Trace Execution box.
Single Step Trace	This command introduces a pause after each script line is executed; this allows you to examine each line in a trace window before you continue.
Report Errors in Expressions	This causes the script, whether tracing or not, to report every error in an expression. These include syntax errors, such as missing or mismatched parentheses, and mismatched variable types.

See **Single Step Trace**.

Scripts Do Script Command

The grey **Do Script:** heading is followed by all the scripts you have chosen to list in the menu. Pull down the Scripts menu and select the script you want to invoke from the list below the **Do Script:** heading.

If you have more script names than will fit in the Scripts menu, the menu will change to read **Do Script...** and is no longer grey. Choosing this command will show a list box of script names. Select a script and choose **OK** to begin script execution. The **Cancel** button will return you to the terminal window without executing any script.

Scripts Step Command

When you run a script in the Trace, you have the option to pause the script after it executes each line. Just check the **Single Step Trace** box in the Trace Script dialog box. Then, when the script begins to run, a **step** button will appear to the right of the Abort Script button.

The **Single Step Trace** command introduces a pause after each script line is executed; this allows you to examine each line in a trace window before you continue.

Scripts Standard Command

This dummies the Scripts Standard command.

Scripts Modem Command

This dummies the Scripts Modem command.

Scripts Type Command

This dummies the Scripts Type command.

Scripts User Script Command

This command will start the execution of a script. See Script Editor for information on how to place a script name in the Scripts menu.

See also: Script Do Script.

Scripts User Script Command

This command will start the execution of a script. See Script Editor for information on how to place a script name in the Scripts menu.

See also: Script Do Script.

Scripts User Script Command

This command will start the execution of a script. See Script Editor for information on how to place a script name in the Scripts menu.

See also: Script Do Script.

Scripts User Script Command

This command will start the execution of a script. See Script Editor for information on how to place a script name in the Scripts menu.

See also: Script Do Script.

Script User Script Command

This command will start the execution of a script. See Script Editor for information on how to place a script name in the Scripts menu.

See also: Script Do Script.

Script User Script Command

This command will start the execution of a script. See Script Editor for information on how to place a script name in the Scripts menu.

See also: Script Do Script.

Script User Script Command

This command will start the execution of a script. See Script Editor for information on how to place a script name in the Scripts menu.

See also: Script Do Script.

Script User Script Command

This command will start the execution of a script. See Script Editor for information on how to place a script name in the Scripts menu.

See also: Script Do Script.

Script User Script Command

This command will start the execution of a script. See Script Editor for information on how to place a script name in the Scripts menu.

See also: Script Do Script.

Script User Script Command

This command will start the execution of a script. See Script Editor for information on how to place a script name in the Scripts menu.

See also: Script Do Script.

Scripts Names Command

This is the **Name** of a driver of the type shown in **Script Type**. There are three types of scripts: Settings Scripts, Modem Scripts and Connection Scripts. For each of these Script Types, **Name** is Settings Document Name, Modem Driver Name or Connection Driver Name, respectively.

See Script Manager

Transfer Send File Command

This leads to a directory dialog box where you may select a file to send and a protocol to use. The protocols supported are: Text, XMODEM, YMODEM, and ZMODEM.

To send a file, follow these steps:

Issue the commands necessary to prepare the receiving computer to begin receiving the file using one of the protocols described below.

Choose **Send...** from the Transfer menu. This leads to a directory dialog box where you may select a file to send and a protocol to use.

Select the transfer protocol from the protocol list box. The default protocol is the last protocol selected in the Protocol Settings dialog box, but the **Send** dialog box will remember the last protocol used.

Select the file you want to send and choose **OK**. Transmission should now begin, and the progress of the transfer will be displayed on the screen in a **Progress Box**.

To cancel the file transfer at any time, choose the **Abort** button in the **Progress Box**. If you or the remote computer cancel the transfer, the transmission of the file will stop. A message will appear on the screen to inform you who canceled the transfer.

IMPORTANT:
If the file doesn't transfer properly, you may have to adjust some of the protocol settings. However, MicroPhone II's default settings will work fine for almost all transfers. These settings are in the Protocol Settings dialog under the Settings menu.

See also: Send Text
Send XMODEM
Send YMODEM
Send ZMODEM

Transfer Send File Text Command

For most text transmissions, MicroPhone II's default settings will be appropriate. However, should you need to change settings, select Text Transfer**...** in the **Settings Menu**. Check the appropriate parameters for: X-on/X-off pacing, Wait for Echo, Wait for Prompt Char, Delay between Characters, Delay between Lines and Word-wrap Outgoing Text at.

Use this command to send plain text files only. If you wish to send a formatted file created by a word processing program such as Microsoft Word, or a page layout file generated by PageMaker, use one of the following protocol commands:

See also: Transfer Send
Send XMODEM
Send YMODEM
Send ZMODEM

Transfer Send XMODEM Command

This is the most commonly used protocol for file transfer. You may use this command to send any PC file or program, including word processing, spreadsheet, page layout, graphs, outlines and pictures.

Check that your protocol settings under Protocol Transfer... in the **Settings Menu** match those of the remote computer. Select XMODEM as the protocol and make the appropriate CRC, 1K Blocks and timeout choices. The default settings will work for most transfers.

There are many combinations of the settings for XMODEM transfers. Some telecommunication services erroneously label these as independent XMODEM protocols, or even YMODEM. MicroPhoneII is able to recognize these variations and automatically changes its settings to match. (Refer to the discussion on cascading in the User's Guide Appendix on File Transfer Protocols.)

See also: Transfer Send
Send Text
Send YMODEM
Send ZMODEM

Transfer Send File YMODEM Command

This protocol is a variant of XMODEM. It allows you to send multiple files, and always sends the file names as part of the transfer. When you choose YMODEM, in the dialog box, additional options appear in the lower left corner.

> Choose **Batch** to send a batch of files. The path will change to show only batch files ("*.MBT"). To create a batch, use the Batch... command.
>
> Choose **File** to send a single file. This changes the path to show all files, which will allow you to choose the file you wish to send.

When sending a batch, you should watch to ensure that the receiving computer issues the appropriate command to receive files if it lacks an auto-detect capability similar to MicroPhone II s.

The program will skip those files it can't find. If you or the remote cancel the transfer, no more files will be sent beyond those already successfully transferred.If you want to send a file using YMODEM-G, simply choose YMODEM from the protocols list. YMODEM-G is the choice of the receiver, so as long as the receiver is set to receive using YMODEM-G, that is how the transfer will occur. (See the Receive... command for more information on YMODEM-G.)

IMPORTANT:
Before using YMODEM for sending files, be sure the remote system supports it. Also, be sure that the names of the files you are sending adhere to the conventions adopted by the remote.

The YMODEM protocol always uses a packet size of 1K and CRC error checking.

See also:
- Transfer Send
- Send Text
- Send YMODEM
- Send ZMODEM
- Transfer Batch...

Transfer Send File ZMODEM Command

In ZMODEM transfers, if a transfer is broken or interrupted, it will remember where that transfer was stopped and can resume transmission with the very next byte at a later date. When you choose ZMODEM from the protocols list, a number of options become available.

Choose Batch to send a batch of files. The path will change to show only batch files ("*.MBT"). To create a batch, use the Batch... command.

Choose **File** to send a single file. This changes the path to show all files, which will allow you to choose the file you wish to send.

If you choose **Text** instead of **Binary**, the receiving computer will be notified that this is a text file. The receiving computer is then able to translate the text file so that it may be read by text editors on that computer.

The program will skip those files it can't find. If you or the remote cancel the transfer, no more files will be sent beyond those already successfully transferred.

Resurrecting ZMODEM Transfers

Interrupted transfers may be resurrected, provided the ZMODEM implementation on the receiving machine supports this feature. As the sender, you merely reestablish the connection and issue the same send command. Then, depending upon the instructions from the receiving computer, MicroPhone II will start the transfer either at the beginning of the file or where the original transfer was left off.

IMPORTANT:

ZMODEM is fast because the sender does not have to wait for acknowledgements from the receiver. When you set up ZMODEM in the Protocol Transfer settings dialog box, you can indicate how often, if at all, MicroPhone II should wait for acknowledgements from the receiving computer. The only time that ZMODEM will resend a block of data is if it receives a negative acknowledgement (NAK) from the remote system.

ZMODEM can be used with certain high-speed modems with built-in error checking, such as Telebit's TrailBlazer, to achieve file transfer speeds exceeding 18,000 bps, depending on line quality.

See also: Transfer Send
Send Text
Send XMODEM
Send YMODEM
Transfer Batch...

Transfer Send Batch Command

This command invokes the batch editor to create a group of files for batch transfers. You'll be presented with a dialog box asking which batch file you want to edit.

A batch file may be created in any text editor; however the file must have the extension "**.MBT**" to be recognized by MicroPhone II automatically. If you create your batch with MicroPhone II's batch editor, the program handles the pathnames and the extension for you.

See XMODEM, YMODEM, and ZMODEM.

Transfer Send Batch Text Command

A batch file can be created (default file extension .MBT) to send multiple files.in text mode

See Send File Text and Batch...

Transfer Send Batch XMODEM Command

A batch file can be created (default file extension .MBT) to send multiple files.in XMODEM mode.

See XMODEM and Batch...

Transfer Send Batch YMODEM Command

A batch file can be created (default file extension .MBT) to send multiple files.in YMODEM mode.

See YMODEM and Batch...

Transfer Send Batch ZMODEM Command

A batch file can be created (default file extension .MBT) to send multiple files.in ZMODEM mode.

See ZMODEM and Batch...

Transfer Receive Command

Since XMODEM, YMODEM and YMODEM-G are all part of the same family of protocols, the method for receiving files works the same for each. If you are receiving a batch of files using YMODEM, you have only to initiate the transfer for the first file; MicroPhone II will automatically start the transfer of all later files. If you are receiving files using ZMODEM, you don't even have to issue a receive file command; the transfer happens automatically.

To receive a file, follow these steps:

1. Instruct the sending computer to begin sending the file using one of the protocols described below.

2. Choose **Receive...** from the Transfer menu. This leads to a directory dialog box where you may select a directory in which to save the file and a protocol to use.

3. Select the transfer protocol from the list. The default protocol is the last protocol selected in the Protocol Settings dialog box, but the Receive dialog box will remember the last protocol used.

4. Choose **OK**. Transmission should now begin, and the progress of the transfer will be displayed on the screen in MicroPhone II's **Progress Box**.

5. To cancel the file transfer at any time, click the **Abort** button in the **Progress Box**. If you or the remote computer cancel the transfer, the transmission of the file will stop. A message will appear on the screen to inform you who canceled the transfer.

IMPORTANT:
If the file doesn't transfer properly, you may have to adjust some of the protocol settings. However, MicroPhone II's default settings will work fine for almost all transfers. These settings are in the Protocol Settings dialog under the **Settings menu**.

See also: Receive XMODEM
Receive YMODEM
Receive YMODEM-G
Receive ZMODEM

Transfer Receive XMODEM Command

This is the most commonly used protocol for file transfer. You may use this command to receive any PC file or program, including word processing, spreadsheet, page layout, graphs, outlines and pictures.

Check that your protocol settings under Protocol Transfer... in the **Settings menu** match those of the remote computer. Select XMODEM as the protocol and make the appropriate CRC, 1K Blocks and timeout choices. The default settings will work for most transfers.

There are many combinations of the settings for XMODEM transfers. Some telecommunication services erroneously label these as independent XMODEM protocols, or even YMODEM. MicroPhone II is able to recognize these variations and automatically changes its settings to match. (See Appendix on File Transfer Protocols in the User's Guide.)

Filenames are not sent with the file during an XMODEM transfer. Consequently, you will be presented with a dialog box asking you to name the transmitted file at the conclusion of the transfer.

See also: Transfer Receive Command
Receive YMODEM
Receive YMODEM-G
Receive ZMODEM

Transfer Receive YMODEM Command

To receive files using YMODEM, simply choose it from the protocols list. It is even possible to receive files using YMODEM when you have chosen XMODEM as the receiving protocol. Because YMODEM is the choice of the sender, MicroPhone II will automatically change to that protocol even if you have chosen XMODEM.

If a batch of files is being received, MicroPhone II will receive all the files into the directory specified for the first one.

Files received using YMODEM will be saved using the names sent along with them. If no name is sent, or if the name does not follow PC filename conventions, MicroPhone II gives the file a temporary name (ending with ".TMP") that you may change later.

YMODEM, an extension to the XMODEM protocol, offers two important benefits:

1. It allows you to send and receive a group of files called a batch.

2. It transmits pertinent information, such as the name and size of the file, along with the data.

The YMODEM protocol always uses a packet size of 1K and CRC error checking.

See also: Transfer Receive Command
Receive XMODEM
Receive YMODEM-G
Receive ZMODEM

Transfer Receive YMODEM-G Command

YMODEM-G is a variant of YMODEM. It provides the fastest file transfers at the price of canceling the transfer on the first error. (Other protocols will try to correct the error.) Consequently, it should be used only with reliable connections such as those provided by error correcting modems. (V.42 and MNP are two common error correction modem standards.)

To receive a file using YMODEM-G, choose YMODEM-G from the protocols list. YMODEM-G is the choice of the receiver. As long as the sender is using YMODEM, the transfer will occur using YMODEMG.

See also: Transfer Receive Command
Receive XMODEM
Receive YMODEM
Receive ZMODEM

Transfer Receive ZMODEM Command

ZMODEM transfers are automatic; you do not have to issue a receive file command. However, if the remote computer does not send the ZMODEM auto-start sequence, you may initiate the transfer by choosing ZMODEM from the protocols list.

Normally, the sending computer initiates the transfer by issuing the appropriate commands. MicroPhone II will detect that a file is coming in and will perform the appropriate formatting. The progress of the transfer will be displayed and files will be saved using the names sent with them.

See also: Transfer Receive Command
Receive XMODEM
Receive YMODEM
Receive YMODEM-G

Transfer Receive Text Command

This dummies the Transfer Receive Text (**Not V2.**)

Transfer Batch Command

A batch is a text file that lists the names of files, with their pathnames, to be sent as a group, using one of the protocols that supports batch transfers: YMODEM, YMODEMG or ZMODEM. A batch file may be created in any text editor; however the file must have the extension "**.MBT**" to be recognized by MicroPhone II automatically. If you create your batch with MicroPhone II's batch editor, the program handles the pathnames and the extension for you.

Choose **Batch...** from the Transfer menu to invoke the batch editor to create a group of files for batch transfers. You'll be presented with a dialog box asking which batch file you want to edit.

To create a new batch, type the name of your new batch in the **File:** text box. If you do not type a file extension with the name, MicroPhone II will append ".MBT" to the file name automatically.

If you do not want any extension added, end the file name with a period, "myfile." for example. Choose **OK** and MicroPhone II will display the batch editor.

Files may be chosen from any directory or disk drive. Also, because a batch file is really just a text file, you can add it (including the one you are creating) to the list of files to be sent.

Click the **Add** button to include selected files in your batch. You may also add a file to the batch by double-clicking the filename.

The **Delete** button removes selected items from the batch list. To remove items from the batch list, select any portion of the line or lines to be removed and choose the **Delete** button. To remove consecutive items, place the cursor on the line of the first item to be removed and choose the **Delete** button. Notice that the item has been removed and the cursor is on the line of the next item. Choose **Delete** again to remove that item. Continue choosing the **Delete** button until all the consecutive items have been removed.

Transfer Select Directory Command

Use this command to name the default directory to receive files. Because you have the opportunity to change this directory each time you receive a file, this setting will apply only to the first time a file is received during the session.

Transfer Display Path Command

This dummies the Transfer Display Path command.

Transfer Insert Point Command

This dummies the Transfer Insert Point command.

Help Index Command

This command brings you to the main index of MicroPhone Help.

Help Keyboard Command

This command brings you to the index of keyboard usage in MicroPhone Help.

Help Commands Command

This command brings you to the index of menu commands in MicroPhone Help.

Help DDE Command

This command brings you to the index of DDE usage in MicroPhone Help.

Help Script Commands Command

This command brings you to the index of script commands in MicroPhone Help.

Help Screen Regions Command

This command brings you to the index of MicroPhone II's screen regions in MicroPhone Help.

Help Using Help Command

This command brings you to the Window help in using the Help Browser.

Help About MicroPhone II Command

This displays the program's name, authors and current version of the MicroPhone II that you are running.

KeySel
HELPID_KEYSEL

Dialog Box Base
HELPID_DLG_BASE

Variable Selection Dialog Box

This dialog box displays a list of all the variables in MicroPhone II that may be linked to another application.

These variables include a number of MicroPhone II's settings functions, as well as any variables created by MicroPhone II scripts.

Script Editor Dialog Box

MicroPhone II's Script Editor provides an environment where you can create and modify scripts.

The text box on the top line is the **Script Name** text box. This is where you name the script.

Below the **Script Name** text box is the **Script Listing Box**. The smaller box in the lower left part of the screen is called the **Command List Box**. The **Script Listing Box** contains the statements that make up the script. These lines will be executed in the order they appear when the script is invoked.

Immediately to the right of the **Command List Box** is the **Variable Name Field** and the **Subcommand List Box**. The **Expression Box** will appear just below these when a script command requires an expression.

(See Script Commands for more information on the commands in the **Command List Box**.)

Invoking a Script

The choices to the right of the script name allow you to assign a script to a Function key, the Scripts menu or a button in the button bar.

> **Function-Key** - In the F-Key box, enter one of F2 through F15, except F10 (depending which are available). This maps the script to the selected F-key on your PC keyboard. Thereafter, pressing this key will execute the script.
>
> **Menu** - Select the Menu box to list the script in the Scripts menu under Do Script. Once generated, the script may be invoked from the menu.
>
> **Button** - Select the .Button box to create a new button for this script. The button will appear at the bottom of the screen, and when you "push" it, script execution will begin. You may display as many buttons as will fit in a single line across the bottom. (Buttons can also be installed and removed from scripts, presenting the user with a hierarchy of buttons.)

Editing a Script in the Script Editor

Below the **Script Listing Box** are the **Paste**, **Copy**, **Cut** and **Clear** buttons. Directly below the **Cut** and **Clear** buttons are two more buttons, **Insert** and **Add New Line**. All of these allow you to edit scripts:

> **Paste** places text from the Clipboard to the left of the text insertion point or replaces any selected text.
>
> **Copy** duplicates selected text and places the copy on the Clipboard.
>
> **Cut** removes selected text from the editor and places it on the Clipboard.
>
> **Clear** erases selected text from the Script Editor.
>
> **Insert** places the command selected in the command list box directly before the line in the script listing box where the cursor resides.
>
> **Add New Line** places the selected command in the command list box after the last command in the script listing box.

The Script Editor also allows you to type text directly to the left of the flashing cursor and to delete text with the backspace key.

Other Buttons

The **OK** button saves the script and returns to MicroPhone II's terminal window. Before leaving the Script Editor, however, the program will check the syntax of the script commands. If an error is found, the Script Editor will remain open and the error will be reported. The OK button is greyed until the script has been named.

The **Check** button initiates a syntax check of the script commands in the **Script Editor**. If there's an error, a dialog box will indicate the line number of the improper command and position the cursor on the line with the error. Otherwise, you will see this alert:

The **Print** button prints the script in your printer's text mode. Be sure that the printer is ready and on-line before choosing this command.

The **Cancel** button returns you to the Script Manager without constructing or modifying the script.

Expression Box

The text box at the bottom of the window allows you to enter text to fully specify the functioning of certain commands. Text is limited to 255 characters.

Alphabetic

You may select this box to rearrange the commands in the **Command List Box** in alphabetical order. Deselect the box to list commands in their original order.

Script Manager Dialog Box

The Script Manager provides access to MicroPhone II's scripting environment. Using the commands in the Script Manager, you can create new scripts, or you can modify, delete, copy, import, export, print and execute existing scripts.

This list box contains all the scripts in the current document specified by Name and Script Type. There are three types of scripts:

1. **Settings** Scripts
2. **Modem** Scripts
3. **Connection** Scripts

Settings scripts are stored in the settings document; these are the scripts used to automate a session with another computer. **Modem** and **Connection** scripts provide instructions for your specific hardware configuration. They are stored in the modem drivers and connection drivers. The difference between these scripts is described in the Editing Drivers chapter in the User's Guide.

Create	Creates a new script.
Modify	Modifies a selected script.
Delete	Deletes selected script(s).
Copy	Copies selected script(s).
Import	Imports a script.
Export	Exports selected script(s).
Print	Prints selected script(s).
Run	Runs a selected script.
Close	Closes the Script Manager.

Scripts Create Script Command

This command opens the Script Editor, where you can create a new script.

Scripts Modify Script Command

This command opens the Script Editor, where your chosen script will be displayed. Follow the guidelines for using the Script Editor to modify the script.

Scripts Delete Script Command

To delete a script or group of scripts, select the script name(s) in the Script Manager and choose the **Delete** button. MicroPhone II will then ask you to confirm the selection, and if you choose the **OK** button in this confirmation box, the script will be deleted.

Scripts Import Script Command

This leads to a directory dialog box where you may select a script to be imported

See also: Script Manager

Scripts Export Script Command

To export a script or group of scripts, select the script name(s) in the **Scripts** list box in the Script Manager and choose the **Export** button.

Scripts Copy Command

To copy a script or group of scripts, select the script name(s) in the **Scripts** list box in the Script Manager and choose the **Copy** button. MicroPhone II will then duplicate each script and append the name of the copy with "copy." If a script with that name already exists, MicroPhone II will add the number 2 to the end.

Scripts Print Command

To print a script or group of scripts, select the script name(s) in the **Scripts** list box in the Script Manager and choose the **Print** button.

Scripts Run Command

To execute a script, select the script name in the **Scripts** list box in the Script Manager and choose the **Run** button.

Scripts Close Command

This command closes the Script Manager.

Communications Settings Dialog Box

This dialog box allows you to select options for setting your modem to work with a particular service, or for preparing for a direct connection.

Modem

Select your modem from the list of available modem drivers. If you wish to use MicroPhone II's default modem settings, you should select Standard, or ensure that none of the drivers are selected.

Install Modem Driver...

The **Install Modem Driver...** button provides access to the files that contain new modem drivers not included in the original MicroPhone II package. Select the new modem driver file (Modem driver files have the extension .MDV.) and choose the Install button. The driver will appear in the list of the installed files at the far right. Any modem driver names associated with the new driver will appear in the **Modem** list when you choose OK.

Connection

Select the means of connection for your communications session from the list of available connections. Your selection should correspond to the port connected to your modem and phone line, or directly to another computer.
MicroPhone II also supports various network communications servers, including those available using Novell, Ungermann-Bass, ComBIOS and DOS device drivers. When one of these connections is chosen from the list, additional fields may appear in the dialog box, requesting more information.

Install Connection Driver...

Just like the **Install Modem Driver...** button, **Install Connection Driver...** provides access to files that contain new connection drivers not included in the original MicroPhone II package. To install a new connection driver file, follow the instructions for installing a modem driver file above.

Baud Rate

Select the correct baud rate for your modem. MicroPhone II's default baud rate setting is 1,200.

Bits per Character

Select one of the number of databits used by your connection from the list. The default choice is 8.

Parity

Select the form of parity checking supported by the remote computer. None is the default choice.

Stop Bits

Stop bits can be set at 1, 1.5 or 2 for any baud rate. The default choice, Auto, will automatically set the appropriate stop bits for the various transmission speeds. Select the setting that is required by the remote computer.

Flow Control

Each computer must be able to regulate the flow of data in case the data is received too fast. MicroPhone II's flow control choices include none, X-On/X-Off software control and hardware handshake. X-On/X-Off is default.

Once all settings have been selected, choosing the **OK** button will save them for the current session and return you to MicroPhone II's terminal window. (If you want to save your settings in a document for use later on, choose the Save Settings as... command from the File menu.)

Choosing the **Cancel** button returns you to the terminal window without changing the communications settings.

Character Set Translation Dialog Box

This dialog box displays a list of alternative character sets that you can use in MicroPhone II. This makes it possible to communicate with systems that replace certain characters in the standard ASCII set with others that are used in European languages, such as ¿ or å or ç. Several ISO-7 substitution sets are provided:

The current character set is selected. To choose another character set, simply select the new one and click **OK**. The default is the ANSI set.

System Preferences Dialog Box

This dialog box allows you to select settings that apply to general MicroPhone II operations.

Size of Screen Buffer

With this option, you can regulate the amount of memory that MicroPhone II reserves for the screen buffer. The size of the buffer is measured in kilobytes. The default is 40K.

If you reduce the size to a smaller number of kilobytes than is currently in the buffer, you will be warned about loss of data.

The smallest acceptable buffer is 3K. At the other extreme, the number you type in the text box should not exceed the size of your machine's available memory.

Display Buffer Full

When the screen buffer is full, additional data that appears in the terminal window will force lines at the top (the first lines that were received) out of the buffer, and these lines will be lost.

With the Display Buffer Full option, you can instruct MicroPhone II to warn you before it begins discarding lines.

The warning message automatically stops data transmission. When this happens, dismiss the warning message by clicking OK. If you want to save the contents of the screen buffer, click on the Pause button to stop the transfer. When transmission stops, select the text you want to save and then choose Save Selection as... or Print Selection... from the File menu. Click the Resume button when the saving or printing is complete.

Yes is the default setting.

Default Document Name

When you start MicroPhone II, the program automatically loads the default settings document. Initially, MicroPhone II's default document is MicroPhone Settings, but you can select any of your settings documents to open at startup. Type the name of your startup document in the text box.

Keep Connection Open

Select Yes if you want MicroPhone II remain connected to the serial port even when it is not in use.

Terminal Settings Dialog Box

This dialog box lets you set the parameters for your terminal.

Terminal Type

The default choice is TTY. Clicking the arrow at the end of the field will display the other terminal types available. These are full-screen terminals; the host computer can send special instructions to change the characters anywhere in the terminal window. Choose the terminal type that best matches that supported by the host computer.

Columns

The columns setting provides a list of common terminal widths. Select the column width that is supported by the remote system. If the required width is not listed, select the columns field and enter the value supported by the remote system. If you set the number of columns to more than will fit on your screen, use the scroll bar at the bottom of the window to scroll left and right. The default choice is 80 columns.

IMPORTANT:
Changing the screen width clears the screen buffer and the screen, so any text that was there will be lost. Thus, you should save important text before changing the screen width.

Rows

The rows setting provides a list of common terminal heights. Select the number of rows supported by the remote system. If you want to change the setting, select the rows field and enter the desired value. Your new value will appear in the list. The default choice is 24 rows.

Font

Terminal window text may be displayed in any font currently installed in your system. However, some full-screen terminals (such as VT102) expect the font to provide special graphics characters. These characters are provided in the DEC VT font. Full-screen terminals also rely on column positions to place text on the screen. MicroPhone II will display all fonts in a monospaced format to support these terminals. DEC VT is the default choice.

Size

Characters can appear using any point size supported by the chosen font. Select the size that you want to see in your terminal window. 9-point is the default choice.

Cursor Shape

This option offers a choice between an underline cursor and a block-shape cursor. Select the shape that you want to see in your terminal window. Block shape is more commonly used and is the default choice.

Backspace Key

This option controls what character is sent when the Backspace key on your keyboard is pressed. Choose between the Backspace character (control-H), the Delete character (Rubout) and Correction (control-H-space-control-H, also known as a destructive backspace). your choice should match the character the host system expects for backspace. The default choice is Backspace.
The Backspace choice is non-destructive. (The remote computer is responsible for removing the deleted characters from your screen.) For any choice, the Delete key on your keyboard will send the other character. (If you choose Correction, your Delete key will not be changed.)

Answerback Message

An answerback message should be used if the remote computer expects to receive a text message when it sends the ASCII Enquire character (control-E, dec 005, hex 05). Type the message in the box; it can be up to 255 characters long. This option is sometimes used by telex systems.

Capture on CR/Capture on Clear

These options allow you to choose how information is saved to a capture file when using a full-screen terminal. If you select Capture on CR, upon receiving a new line control sequence, data will be captured to a capture file (if open), or printed to the printer (if on). If you select Capture on Clear, upon receiving a "Clear Screen" escape sequence from the remote, data will be captured to a capture file (if open), or to the printer (if on). Capture on CR is the default; it is also automatically chosen when using a line based terminal (TTY).

Strip 8th Bit

This option gives you some control of how incoming characters are interpreted. If you deselect this option, MicroPhone II will accept characters from an extended, or eight-bit, character set. Since many systems use only the 7-bit ASCII set, Strip 8th Bit is the default choice.

Local Echo

With Local Echo, MicroPhone II displays characters in your terminal window as you type them. Without **Local Echo**, MicroPhone II does not display your characters unless the remote computer supports echoplex, which will echo them back to your screen. The default is Local Echo off. Check the box if you wish to set Local Echo on.

New Line

New Line should only be selected when the remote computer expects line feeds following carriage returns. (Normally, MicroPhone II sends only a carriage return when you press the Enter key.) When New Line is enabled, pressing the Enter key will generate a carriage return (CR) followed by a line feed (LF). When a line feed or form feed (FF) is received, it will be interpreted as a carriage return followed by the LF or FF.

The default setting is New Line off.

Auto Wraparound

This option will break a long line, so that information beyond the column width will appear on the next line. MicroPhone II automatically wraps at the appropriate place, depending on your choice of column width. This choice only affects the way information is displayed on your screen. If you are receiving text to a file, that text will be saved with its original line length. The default setting is Auto Wraparound off.

Text Transfer Settings Dialog Box

The settings for text file transfers are made within this dialog box. These settings allow you to structure a transfer so that it will be successful given your particular configuration.

X-On/X-Off Pacing

Check whether or not X-On/X-Off flow control is supported by the remote computer during text transfers. The default setting is to use X-On/X-Off Pacing While Sending and While Receiving. (This setting is ignored if hardware flow control is selected under Communications Settings.)

Wait for Echo

Wait for Echo allows you to control the flow of your text transmission to the remote computer if it does not support X-On/X-Off. MicroPhone II will stop sending data and wait for the appropriate echo before resuming transmission.

If **None** is selected, MicroPhone II sends data without waiting for echo at all. When communicating with a remote that can't echo characters, your setting must be None.

If **CR** (carriage return) is selected, MicroPhone II sends one line at a time, waiting for a carriage return from the remote before sending the next line.

Selecting **LF** (line feed) is identical in operation to **CR**, except that an echo of the line feed character triggers sending the next line
.
The **All** choice should be selected as a last resort, because it slows down transmission considerably. This choice sets the program to wait for the echo of every character that it transmits. It is also somewhat risky; if any of the characters gets garbled during transmission, the transfer will fail.

The default choice is None.

Wait for Prompt Char

This option allows you to pace your text transfer to the remote computer. It works only with systems that are able to prompt for each line and that provide the same character for each line prompt. When you type a character in the box, MicroPhone II will send one line, then wait for the prompt character before sending the next. The default setting here is no prompt character. (Leave the box empty.)

Delay between Chars
Delay between Lines

When connected to a system that cannot do X-On/X-Off pacing, does not echo characters and can't provide prompts, you may wish to introduce time delays to simply slow down transmission from MicroPhone II, allowing the other computer to keep pace. Set the delays to occur after every character you send, or after entire lines.

Delays are in 1/60th second increments. If you enter 10 in the Delay between Chars text box, MicroPhone II will pause 1/6th of a second between each character that it sends. If you enter 30 in the Delay between Lines text box, MicroPhone II will send one complete line every half second. The default setting is 0 for both of these.

Word-wrap Outgoing Text

If the text you are sending has longer lines than the width of the remote station's screen, you can set Word-wrap Outgoing Text to the appropriate width. Enter the number of characters per line that the

remote computer can display. If you do not want any wraparound, set the Word-wrap value to zero. The default is 80 columns.

End Outgoing Lines with

You can end outgoing lines with a **CR** (carriage return), **LF** (line feed), **CR LF** (carriage return, then line feed), or **LF CR** (line feed, then carriage return). These allow MicroPhone II to communicate with almost any computer, whether or not it expects to supply its own carriage returns or line feeds. You may also select **Nothing**. The default selection is **CR LF**.

End Captured Lines with

The choices for this option - **CR** (carriage return), **LF** (line feed), **CR LF** (carriage return, then line feed), **LF CR** (line feed, then carriage return), or **Nothing** - will determine the characters appended to the end of each line of incoming text saved in the capture file. The default selection is **CR LF**.

The **OK** button accepts the changes made in the Text Transfer Settings dialog box and then returns to MicroPhone II's terminal window.

The **Cancel** button returns to the terminal window without any changes to text transfer settings.

Protocol Transfer Settings Dialog Box

This is the dialog box where you may change the settings for MicroPhone II's file transfer protocols.

Default Protocol

This option provides a list of protocols supported by MicroPhone II. When you select a protocol from this list, you are presented with specific options for that protocol. The last chosen protocol becomes the default protocol in the Transfer menu's Send and Receive dialog boxes.

Timeout

Timeout allows you to set the protocol transfer timeout (not less than 10 seconds). This setting is used for all protocol transfers. The choice has no effect on the timeout used by the remote.

File transfers using XMODEM, YMODEM and ZMODEM generally require that no more than 10 seconds elapse between sending a block and receipt of an acknowledgement. However, some long distance connections, such as a satellite link, may require a longer timeout.

XMODEM

When the XMODEM protocol is selected, you may specify information about block size and the type of error control. These options are discussed below.

CRC

MicroPhone II will automatically default to the checksum method of error control if **CRC** is not supported by the remote computer. However, if the remote does not support **CRC**, turning this option off in advance (switching to checksum) may save up to 30 seconds on transmission time (the time it takes for MicroPhone II to switch to checksum).

If you know in advance that the remote computer does not support XMODEM CRC protocols, then select **CRC** No. The default is Yes.

1K Blocks

File transfers using 1K Blocks (1,024 bytes) rather than 128 byte blocks can significantly reduce file transfer time because there are fewer blocks to acknowledge. When 1K Blocks is set to its default value, **Automatic**, PC to PC file transfers with MicroPhone II at both ends automatically use 1K blocks. With other remote systems, MicroPhone II can detect a 1K capability and switch to this mode.

The three 1K Blocks options work as follows:

Automatic
Sending files: If the remote signals that it can receive 1K blocks (1K signal), MicroPhone II will automatically use this method. Otherwise, 128 byte blocks will be sent.
Receiving files: MicroPhone II will send a 1K signal but will be ready to receive 128 byte blocks if the remote lacks 1K capability.

On
Sending files: MicroPhone II will force a 1K transfer. This should be chosen when the remote can receive 1K blocks but does not send a 1K signal.
Receiving files: As in **Automatic**.

Off
Sending files: MicroPhone II will send 128 byte blocks whether or not the remote supports 1K

blocks.

Receiving files: MicroPhone II will not send a 1K signal. It will detect and receive 1K blocks if sent by the remote.

YMODEM

CRC and 1K blocks are standard for **YMODEM**. You may specify the **Timeout** (described above).

ZMODEM

When you select **ZMODEM** from the protocol list, the other options in the dialog will change to the settings available for the **ZMODEM** protocol:

Packet Size

The ZMODEM protocol allows you to designate the size of the packets. The default is to divide a file into packets of 1K (1024 bytes), but before a ZMODEM transfer begins, the two machines involved "compare notes" about their requirements. If the remote machine can only transfer 128 byte packets, MicroPhone II will automatically reset itself to match that computer. You shouldn't need to change the default unless your transfer continually aborts due to bad data errors.

The largest packet size MicroPhone II supports is 1K.

Window Size

In **ZMODEM**, you can designate the window for errors - in other words, you can tell MicroPhone II how many bytes should be sent before pausing for an ACK (acknowledgement that those bytes were received correctly).

To facilitate speedy transfers, the default is 0, which means that the program will never wait for an ACK, but will respond only to a NAK (negative acknowledgement) by resending the packet that was not received correctly.

If you do change the window size, it should be a multiple of the packet size. For example, 8192 bytes would be an appropriate window size if your packet size is 1024 bytes.

CRC Type

The default error control is CRC-32, a 32-bit CRC. Older systems may use a 16-bit CRC. When MicroPhone II is set for CRC-32, it will automatically drop down to CRC-16 if the remote system requires it.

Auto-detect

The Auto-detect choice allows MicroPhone II to automatically receive ZMODEM transfers. That is, you do not have to issue a receive file command - the file transfer begins as soon as the remote issues its send file command.

Resurrect Transfers

This option allows you to resume an aborted file transfer without loss of time. When MicroPhone II receives a file with the ZMODEM protocol, the program stores information about the transfer. If the transfer is interrupted, MicroPhone II will remember exactly how much of the file was successfully transmitted. At a later date, the transfer may be resumed right where it left off.

Escape Control Characters

If the file you are transmitting contains control characters, you may want to select this option. Escape Control Characters temporarily masks control characters during a ZMODEM transfer so that those characters will not affect the transfer (by sending an X-Off, for example). The default is Escape Ctrl Chars off.

Startup Action Dialog Box

This dialog box allows you to set MicroPhone II to automatically dial a service or run a script each time this document is opened.

The None, Dial Service and Do Script option buttons control what startup action, if any, is initiated. Only one action may be set at a time.

Turnkey Startup

You may set the program to conduct an entire communications session without your intervention. MicroPhone II can run a script automatically as soon as it is loaded. To accomplish this, follow these steps:

Choose System Preferences from the Settings menu.

In the **Default Document Name** text box, enter the name of the settings document that you want automatically opened when MicroPhone II is loaded.

If the script you want is not already in the default document, use the Import Script ... command under Edit Scripts in the Scripts menu to import it to this settings document.

Choose Startup Action... from the Settings menu, and select your startup script.

When MicroPhone II is started up directly, it will always open the default settings document named in the System Preferences dialog box. Once this document is loaded, the Startup Action invokes the designated script and the session begins.

Watch Me Dialog Box

In the **Watch Me** dialog box, you can record a communications session, and you can edit the scripts generated by **Watch Me**.

Record a telecommunications session

Enter the script's name in the **Name** text box.

If you want to be able to invoke your script with a function key, bring the focus to the **F-Key box** and press the key (between F2 and F15, but not F10, depending how many are on your keyboard).

Check the **Menu** box if you want the script name to be listed in the Scripts menu under Do Script.

Check the **Button** box if you want to install a button bearing the script's name in the button bar.

Move the focus outside the **Watch Me** editor to the terminal window, and conduct the session as you normally would. You can see a script being generated in the **Watch Me** editor while you work.

Choose **Close** to save the new script when you are at a point where you want to end your **Watch Me** script.

Choose **Cancel** to remove **Watch Me** from the screen and return to the terminal window without saving the new script.

Editing a Script in Watch Me

You may also edit a script that is still in the open Watch Me editor. This simple text editor allows you to type text to the left of the text insertion point (blinking vertical line) and to delete text using the backspace key. In addition, the **Paste**, **Copy**, **Cut** and **Clear** commands are available as screen buttons and allow for the following edits:

Paste places text from the Clipboard to the left of the text insertion point or replaces any selected text.

Copy duplicates selected text and places the copy on the Clipboard.

Cut removes selected text from the editor and places it on the Clipboard.

Clear erases selected text from the Watch Me editor.

Trace Script Dialog Box

Use this dialog box to set the trace options. When a script runs, you'll see a box, or a series of boxes, that shows which script line is about to execute.

To use Trace, check the **Trace Execution** box.

Single Step Trace

When you run a script in the Trace, you have the option to pause the script after it executes each line. Just check the **Single Step Trace** box in the Trace Script dialog box. Then, when the script begins to run, a step button will appear to the right of the Abort Script button.

The Single Step Trace command introduces a pause after each script line is executed; this allows you to examine each line in a trace window before you continue.To continue to the next line of the script, click the step button.

Report Errors in Expressions

This causes the script, whether tracing or not, to report every error in an expression. These include syntax errors, such as missing or mismatched parentheses, and mismatched variable types.

File Dialog Box

This dialog box allows you to access a MicroPhone II document by **Name** or by **File**.

> A **Name** identifies a MicroPhone II document without the restrictions imposed by DOS for identifying files. Names may be any length and contain any sequence of characters, including spaces and punctuation.
>
> A **File** identifies a MicroPhone II document using the DOS filename.

You can access a document by Name only when opening and saving MicroPhone II documents. The directory dialog box is used to manipulate files. The File option button and the Name option button is used to toggle between the two dialog boxes.

Names Dialog Box

Enter a document name in the Name text box and choose **OK**.

> The **All Names** list box allows you to select any of the document names that are installed. Selecting a name in this list will enter the name in the Name text box (described below).
>
> The **File** option button changes the contents of the names dialog box to how the directory dialog box (described below). A filename may be entered in the File text box without choosing the option button. MicroPhone II will act on the file if it is in the current directory.
>
> The **Path** identifies the file holding a selected document name. It also identifies where a document will be saved. (See the directory dialog box discussion below to change the path.)
>
> Once a document name is selected, it automatically appears in the **Name** text box. To choose a document name that is not listed in the Names list box, move to the Name text box and type the document name.
>
> When a document name is entered in the Name text box, the **OK** button will darken. Click it to complete the command with the selection.
>
> The **Cancel** button returns you to the MicroPhone II terminal window without completing the command.

MicroPhone II's names dialog box also includes other controls specific to the command which invoked it. These controls are described in the topics describing the invoking command.

Directory Dialog Box

Choosing the **File** option button in the names dialog box shows the directory dialog box. Here you may change the directory and filename used to hold the MicroPhone II document. Select the proper disk drive, directory and filename, and then choose **OK**.

> The **Drives** list box allows you to select any of the disk drives either on your computer or connected to your computer over a network.
>
> The **Directories** list box initially lists the subdirectories of the open directory. To enter a subdirectory, select it. To exit from a subdirectory, choose the double dots (. .).

The filenames in the **Files** list box are those in the selected directory that contain the extension specified by the path.

The **Path** filters the files shown, so that only the files which you specify will appear in the Files list box. To change this filter so that this list box will display other files, simply type the new filter in the File text box and press the Enter key. The wildcard characters ? and * may be used.

Once a filename is selected, that filename automatically appears in the **File** text box. To choose a file that is not listed in the Files list box, move to the File text box and type the filename. If the specified file is not in the shown directory, you must provide the path to the file.

The **Name** option button changes the contents of the directory dialog box to show the list of MicroPhone II document names that have been installed (described above).

When a filename is selected, the **OK** button will darken. Click it to select the file.

The **Cancel** button returns you to the MicroPhone II terminal window without completing the command.

MicroPhone II's directory dialog box also includes other controls specific to the command used to invoke the dialog box. These controls are described in the topics which explain the invoking command.

DDE Messages

In order to send or receive DDE messages with MicroPhone II, you should first become familiar with the DDE terminology. The application initiating the DDE conversation is called the **client**. The receiving application is called the **server**. In addition, DDE messages are built from three parts: **application**, **topic** and **item**.

Application identifies the recipient of the DDE message (server) by its DDE name. A windows application that supports DDE will usually use its Windows name, or a shortened version, as its DDE name. For example, MicroPhone II's DDE name is MicroPhone. If there is more than one instance of MicroPhone II running, the topic identifies which instance responds to the DDE message

Topic identifies the subject of the DDE conversation. For most applications, the topic is the name of the document intended to receive the DDE messages, such as a MicroPhone II settings document or an Excel worksheet. However, some applications also support other topics, such as "System," which sends messages directly to the application. MicroPhone II supports the System topic to open a settings document or exit the application. A settings document must be open before any other activities can take place.

If MicroPhone II is open with a settings document other than the one specified by the topic, MicroPhone II will not respond to the DDE message. The initiating application must then launch MicroPhone II with the desired settings document.

Item designates the content of the message sent by the client application. This may be any DDE command supported by the server application, such as a request to link a MicroPhone II variable to an Excel spreadsheet cell, link all incoming text to a Microsoft Word document, or to execute a MicroPhone II script.

MicroPhone II scripts can send the following DDE messages to another Windows application: Advise, Execute, Initiate, Poke, Request, Terminate and Unadvise. The DDE Ack and Data messages are sent automatically in response to messages from the server application.

Starting Another Application from MicroPhone II

MicroPhone II can send a DDE message to an application only if it is running. To start any application, including COMMAND.COM, use the script command Application * Run or Application * Load.

Responding to Another Application

Any application that supports DDE can request information from MicroPhone II, or even send commands, as long as MicroPhone II is running. MicroPhone II always listens for any DDE messages addressed to it using the "MicroPhone" application name and the name of an open settings document (or "system") as the topic. The one exception is when the DDE Inhibit setting has been selected for that settings document.

IMPORTANT:
Once MicroPhone II is running, it will respond to any DDE message properly addressed to it. DDE messages take priority over all other tasks in MicroPhone II.

If a script is executing when a DDE message is received, that message will execute immediately, paralleling any currently executing MicroPhone II tasks, even if that task is a script or a file transfer. DDE messages can also execute other scripts, supply their own scripts to be executed or even end script execution altogether. Remember, MicroPhone II supports the execution of any script or menu command from DDE, potentially changing the current MicroPhone II environment. Consequently, using DDE haphazardly can cause problems with your MicroPhone II session. Carefully plan your use of DDE to avoid conflicts.

Use DDE with settings documents which are specifically set up for DDE. To employ the power of DDE intelligently and protect your scripts from unexpected results, establish a busy flag in your script. Create a variable such as "busy" using MicroPhone II's Set Variable * from Expression script command and set it to false. Whenever your scripts begin something that should not be interrupted, such as an on-line session, set "busy" to true. Then, when you initiate a DDE conversation with MicroPhone II and the DDE settings document, use the DDE Advise (or similar hot link command) in the initiating application to test the value of "busy". If "busy" is true, don't send any DDE messages until the flag changes to false. Any number of these variables can be used to inform DDE client applications about the current status of a MicroPhone II session.

Getting Data from Another Application

To receive data from another application, you must first initiate a DDE conversation. Do this using MicroPhone II's DDE * Initiate script command. This command must specify a variable name that is used to identify this conversation. Although you can have any number of conversations open at one time, you should terminate any conversations that are not needed in the immediate future. Here's a sample script; an explanation follows:

```
1       DDE * Initiate ServerName "'Excel','System'"
2       Set Variable * status from Expression "'Conversing with Excel'"
3       If Expression "ServerName <> 0"
4               DDE * Execute ServerName "'[OPEN("WORKSHT.XLS")]'"
5               DDE * Terminate ServerName
6       Else
7               Application * Run "'Excel WORKSHT.XLS'"
8       End If
9       DDE * Initiate ServerName "'Excel','WORKSHT.XLS'"
10      If Success
11              DDE * Advise ServerName "'R9C1', 'total'"
12              DDE * Poke ServerName "'R3C1', '7'"
13              Wait Seconds "10"
14              DDE * Terminate ServerName
15      End If
16      Alert OK "'Your new total is ' & total"
17      DDE * Initiate ServerName "'Excel','System'"
18      If Success
19              DDE * Execute ServerName "'[CLOSE(TRUE)]'"
20              DDE * Execute ServerName "'[QUIT( )]'"
21              DDE * Terminate ServerName
22      End If
```

The first line of the script initiates a DDE conversation with Excel. If Excel responds, ServerName is non-zero (true), MicroPhone II asks Excel to open the specified spreadsheet (line 4). (For this example, the spreadsheet must be in the same directory as the MicroPhone II settings document.) This spreadsheet can now be seen as a topic for later DDE commands.

MicroPhone II then terminates that conversation (line 5). If Excel does not respond to the first script line, then the value of the variable ServerName is zero (false). This would happen if Excel is not running, so MicroPhone II sends a command to launch it with the specified spreadsheet (line 7).

Next, the script initiates a new conversation with the specified spreadsheet (line 9). If that is successful, a hot link is set up from the spreadsheet cell in row 9 column 1 to the MicroPhone II variable total. If total does not exist, MicroPhone II will create it. When the value of that cell changes (as the poke command will do), total will be updated automatically as long as the conversation remains open.

MicroPhone II changes the value of the spreadsheet cell in row 3 and column 1 to 7 (line 12). Remember, if this results in a change in the value of cell R9C1, total will be updated in MicroPhone II. For this example, a wait is included to allow time for Excel to update the spreadsheet, keeping the conversation open long enough for the update to occur. Normally, other actions would take place before terminating the conversation and the wait would not be necessary.

The conversation is then terminated (line 14), ending the automatic updating of total set up in line 10. Finally, a new conversation is established with Excel to close the spreadsheet, and save the changes made by the script.

ServerName contains a system generated number to identify a particular DDE conversation. Because

MicroPhone II supports any number of consecutive conversations, the value of this variable is used by other DDE commands to identify which conversation to use. This number will be zero (false) if the conversation is not open. If the conversation is open, ServerName will contain a non-zero value (true). The actual value of ServerName is not important, only that it is not zero.

In this example, ServerName is tested for a non-zero value (line 3) instead of merely testing for success. This is necessary because there is another command (line 2) that changes the value of the success flag between the time of initiating the DDE conversation and the actual exchange of DDE messages.

Normally you would place a command like the one in line 2 before initiating the DDE conversation. Here, it illustrates how you can test for an open conversation even if the scripts logic requires the test at a later time.

Executing Commands in Another Application

Use MicroPhone II's DDE * Execute script command to send a message for execution by the server application. As with all DDE commands, the server is expected to understand the message you send, or the command will fail. Typically, a message that is to be executed is enclosed between brackets, as in the line:

```
DDE * Execute ServerName "'[CLOSE(TRUE)]'"
```

The particular command within the brackets must conform to the syntax required by the server application.

You can also send a list of commands to be executed, such as a script for MicroPhone II. Each complete command is enclosed between brackets, although all are within a single MicroPhone II expression (the double quotes. See Using Quotation Marks). If you send a list of commands to MicroPhone II using the DDE Execute command, they will not be named or saved. Once the execution is complete, the commands are discarded.

Use the Script * Import command to send a list of commands to MicroPhone II using DDE and have them saved. First, create a text file of the script commands using the appropriate syntax. Then, send MicroPhone II the command:

```
DDE * Execute "'[Script * Import ""'newscrpt.txt""']'",
```

where newscrpt.txt is the name of the text file containing the script. Next, send the command:

```
DDE * Execute "'[Settings Save][Do Script * ""'newscrpt""']'"
```

to MicroPhone II. The settings document will be saved and the imported script will be executed.

IMPORTANT:
The client application must ensure that any commands sent to the server follow the syntax required by the server application. The client must also check for error messages sent from the server. If MicroPhone II receives any improper commands, they will be ignored. However, if any portion of the message can be executed, MicroPhone II will respond with a DDE acknowledgement and execute that portion of the message.

Using Quotation Marks

Double quotes identify the bounds of a MicroPhone II script command expression. Single quotes identify literal strings, to differentiate them from variables.

However, if a single quote is part of a literal string, it must be preceded by another single quote. This prevents the imbedded single quote from being misinterpreted as the end of the literal string.

In the example,

```
"'[Script * Import ""newscrpt.txt""]'"
```

the first set of quotes is a double quote followed by a single quote. There is a corresponding single quote followed by a double quote at the end of the expression. This group of quotes identifies the expression and the contents as a literal string.

The Script Editor will automatically enter the outermost pair of double quotes, so you don't have to enter them as part of the expression.

The square brackets within the literal string are part of the syntax for the DDE * Execute command. You must enter these to identify the beginning and end of an executable command.

Between the brackets is the text of a MicroPhone II script command that is going to be sent to MicroPhone II by the DDE * Execute command. This text must appear as it would in the script listing box if the command were built in the Script Editor (the asterisk is not necessary). If the script command between the brackets uses an expression, that expression must be bounded by double quotes, as it is in the above example.

The expression of this imbedded script command may contain a literal string, which must be bounded by single quotes to differentiate it from a variable. However, since the imbedded Script * Import script command is itself part of an expression, a single quote within the expression of the imbedded Script * Import script command will be misinterpreted. It will be incorrectly seen as the right end of a literal string of the parent DDE * Execute script command's expression. Therefore, each single quote that is part of the imbedded expression has another single quote just before it. The pair of single quotes is used to place a single quote within a literal string.

You can also use a variable to represent a string - or part of it - between single quotes. For example, "'[Script * Import ""newscrp' & numbr & '.txt""]'" will build a complete string expression by inserting the value of the variable numbr as part of the filename newscrp*.txt. If numbr is '3', then the filename of the imported script is newscrp3.txt.

See also Script Expressions.

Focusing

When the program is loaded, the focus is at the cursor in the terminal window.

To bring focus to	Press
the Connect or Pause button first	Shift F6
a script button first	Ctrl F6
the cursor	Alt F6

Repeating Shift F6 or Ctrl F6 transfers the focus from the cursor to each of the buttons available at the bottom of the screen and then back to the cursor.

To bring the focus immediately back to the cursor from any button (without passing each button along the way), use Alt F6.

Pushing Screen Buttons

When you change the focus to a screen button, you are effectively selecting that button. Once a button is highlighted (the button has a heavy black border around it), press the spacebar to "push" the button.

To select	Press
a highlighted screen button	spacebar

Switching Windows

You can use the keyboard to move between different windows in MicroPhone II, or move between MicroPhone II and other Windows applications. Use the following key combinations:

To move the focus to	Press
each MicroPhone II window	Alt F6
another Windows application	Alt Esc or Alt Tab
another Windows application in reverse order	Alt Shift Esc or Alt Shift Tab

Editing

Some of the commands in the Edit menu may also be issued with the following key combinations:

To	Press
Copy	Ctrl Ins
Cut	Shift Delete
Delete	Delete
Paste	Shift Ins

Control Menu

The commands in MicroPhone II's Control-menu may also be issued with the following key combinations:

To	Press
Restore	Alt F5
Move	Alt F7
Size	Alt F8
Minimize	Alt F9
Maximize	Alt F10
Close	Alt F4

See also Command Menus.

Command Menus

Some commands in the menus have unique key sequences that will activate them directly from the keyboard. These key sequences are shown in the menu to the right of the command. Additionally, any menu command can be activated using the following method:

1. Bring the focus to the menu bar by pressing the Alt or F10 key.

2. Open one of the menus by pressing the letter that is underlined in the name of the menu you want to open. (Use the space bar for the Control-menu.)

   Alternatively, you can use the arrow keys to move the selection bar to the menu you want to open. Press the down arrow key to open the selected menu.

3. Select the menu command you want to activate by pressing the letter that is underlined in the name of the command.

   Alternatively, you can use the arrow keys to move the selection bar to the command.

4. Activate the selected menu command by pressing Enter.

All drop-down menus will remain on the screen until a command is chosen, or until either the Alt or the Esc key is pressed to dismiss them.

Action	**Key Sequence**
Move focus to the command menus	Alt or F10
Move focus back	Alt or F10 again.
Open a command menu (when focus is on the menu bar)	underlined letter or arrow keys
Select a menu item (when focus is on the menu bar)	underlined letter or arrow key
Activate the selected menu item	Enter
Close a command menu	Alt or Esc

See also Control Menu.

Dialog Boxes

Each section of a dialog box contains a field in which you can enter information. When a dialog box first appears on the screen, the focus is at the first field. To move to another field, use one of the following methods:

To move to	Press
the next field	Tab
the previous field	Shift Tab
a specific field	Alt + underlined letter.

Fields in a dialog box are normally one of five types: a **command button**, a group of **option buttons**, a **check box**, a **list box**, or a **text box**. Each type has a different method of accepting input from the keyboard.

Command Button
Once a command button (usually Cancel or OK) has been selected with the Tab key, issue that command by pressing either the Enter key or the spacebar.

Option Buttons
When you tab to a group of option buttons, the focus is on the button that was previously selected. To select another option button, press one of the arrow keys until the button you want is highlighted. Then tab to the next field.

Check Box
Each check box is considered a field. To check a box, first select it using the Tab key, then press the spacebar. For check boxes, the spacebar acts like a toggle switch that either enters or removes an "x" from the box.

List Box
Once your focus is on a list box, either press the up or down arrow keys to scroll through the box and make a selection, or press the first letter of the listed item you want. When the item has the focus (has a dotted line around it), press the spacebar to select it.

Text Box
You know you are in the text box field when a blinking cursor appears inside the box. Characters that you type will appear to the left of this insertion point. Inside a text box, the Backspace, Delete and arrow keys are functional.

Action	Press
Choose a command button	spacebar or Enter
Select an option button	arrow keys
Check a box	spacebar
Highlight an item in a list box	arrow keys or first letter

Choose a list box item	spacebar
Enter information in a text box	type the information

Scrolling and Selecting Data

Action	Press
Change to Key Select Mode	Scroll Lock
Scroll	arrow keys
Select data	Shift, arrow keys
Remove highlighting (Key Select mode)	arrow keys
Remove highlighting (terminal window)	Scroll Lock (twice)

Key Select Mode

MicroPhone II has a special mode for scrolling and selecting data from the keyboard: **Key Select Mode**. To set the terminal window in this mode, press the Scroll Lock key. The words "KEY SELECT MODE" appear at the right of the menu bar; the script button area will darken, and the cursor will change to a vertical bar. To return to "terminal window mode," press Scroll Lock again.

In Key Select Mode, you can use all of the direction keys - Home, End, PgUp, PgDn and the arrow keys - to move the insertion point to any character in the work area or screen buffer.

IMPORTANT:
Use Key Select Mode only to select data for saving, printing or copying. In this mode, you cannot type, erase, copy or paste data, nor can you choose screen buttons. In addition, no data will be received or displayed. Although many of the menu commands are still available, it's best to toggle back to "terminal window mode" before you use any of them.

Selecting Data

In order to use the Save Selection as... and Print Selection commands from the File menu, or the Copy and Paste commands from the Edit menu, you must select the data upon which these commands will act. Use the following procedure to select data:

1. Press the Scroll Lock key to change to Key Select Mode.
2. Use the direction keys to position the cursor just before the first character or just after the last character that you want in your selection.
3. Hold down the Shift key and use any of the direction keys to begin the selection. While you are holding down the Shift key, each press of a direction key will extend the selection.
4. To end the selection process, release the Shift key. Pressing another direction key will remove the highlighting.
5. Once a block of data has been selected, press the Scroll Lock key again and then issue your command.

Break Keys

Breaks are used to send signals to the remote computer by changing the line conditions. From scripts, send a break by employing the Signal command.

From the terminal window, use the following key sequences:

To send a	**Press**
short break (233 milliseconds)	Ctrl-break or Alt `
long break (3.5 seconds)	Ctrl-Shift-break or Alt ~

Control Characters

In the Script Editor's expression box, control characters are entered as a caret (^) followed by the corresponding letter. The caret indicates that the next character should be interpreted. For example, ^M is used for carriage return. To type a control character, hold down the Ctrl key and press the appropriate letter key.

The table below shows how to indicate some of the important control characters to MicroPhone II.

To indicate	Type in Script Editor	Appears as
Backspace	Ctrl H	^H
Caret	^	^
Carriage Return	Ctrl Enter	^M
Carriage Return, Line Feed	Shift Enter	^M^J
Escape	Crtl [	^[
Form Feed	Ctrl L	^L
Indent	Ctrl Tab	^I
Line Feed	Ctrl J	^J
Quotation mark	"	^"
RS (dec 030, hex 1E)	^^ or Ctrl ^	^^
X-On	Crtl Q	^Q
X-Off	Ctrl S	^S

To send any of the available control characters of the ASCII Character Set from the terminal window, hold down the Ctrl key and press the associated character.

Within scripts, control characters are considered to be text strings and so must be enclosed in single quotes; the caret indicates that the next character should be interpreted.

To send a literal string which is equivalent to the characters used for control characters, type the characters as a concatenated string: '^' & 'M'. Or, use the Char function if you want to send characters, bypassing MicroPhone II's control character interpreter.

See also Script Editor

Title Bar

The band along the top of MicroPhone II's terminal window contains the program name, a hyphen and the name of the settings document currently in use. When you first load the program, it will open with the default MicroPhone II settings document, MicroPhone Settings. A new settings document is indicated by the word (Untitled).

Maximize/Minimize Boxes

The two boxes at the right of the title bar change the size of the MicroPhone II window. The up arrow will enlarge the window to fill the screen, and the down arrow will reduce it to an icon so that you may easily view another application. After clicking on the Maximize arrow, a pair of arrows appears in the far right corner. Clicking on this pair of arrows will return the window to its default size.

Control-menu

The Control-menu can be accessed from the Control-menu box, left of the title bar. This drop-down menu contains commands that can move, close and change the size of your MicroPhone II window. Notepad, Windows' text editor, is also available from this menu.

Menu Bar

The menu bar is directly below MicroPhone II's title bar. The drop-down menus provide access to all the MicroPhone II commands from the main screen of the program.

Terminal Window

When you open the MicroPhone II application, the main window that you see is the terminal window. It displays all the data received from a remote computer, as well as commands and responses that you have typed.

As characters enter the terminal window, it may become full and the lines will begin to scroll. The older lines become part of the screen buffer. The terminal window contains the newest lines (usually the "bottom" 24 lines). Together, all the data in the screen buffer and terminal window is known as the session data.

Text, whether typed, pasted or sent by the host computer, enters the terminal window at the insertion point. The cursor, which resembles a block or an underline, identifies the location of the insertion point. The cursor is usually positioned directly after the last character that was sent or received. However, with VT102 and other full-screen terminals, the host computer may move the cursor anywhere in the terminal window.

Don't confuse the cursor and insertion point with the I-beam pointer. The I-beam is a mouse pointer that you can use to select a block of text anywhere in the terminal window or the screen buffer.

Scroll Bars

The vertical and horizontal scroll bars frame the right and bottom sides of the work area. They contain arrows at either end for repositioning your view of the screen and the screen buffer. The scroll box that moves inside each bar indicates your place in relation to the length and width of all the information in the buffer. The bottom scroll bar is white when no lines extend beyond the width of the screen.

Button Bar

The button bar provides one way to initiate script execution - with an on-screen button in the area just below the horizontal scroll bar. Once you've assigned a button to your script, just click it, and MicroPhone II will execute the script.

Pause/Resume Button

When you click the Pause button - left of the horizontal scroll bar - MicroPhone II alerts the sender to stop sending information, and the button automatically changes to read Resume. If you click Resume, the program will then alert the sender to continue the flow of incoming information, and this button will again read Pause.

Connect Button

The Connect button may appear to the left of the horizontal scroll bar. This button indicates the connection to your serial port is not yet open. (A closed connection allows other applications to use it.)

To open the connection to your serial port, choose the Connect button. You can then type directly to your modem or whatever is connected to the serial port. This is the same as choosing Connection Open from the Phone menu. The Connect button will change to Pause and the Connection Open in the Phone menu will change to Connection Close. To close the connection, choose Connection Close from the Phone menu.

Abort Buttons

MicroPhone II gives you the option to terminate the execution of certain commands - Dial Service, Execute scripts, Print, Send and Paste - after they have started. As these commands begin to function, an abort button appears at the bottom of the work area, just to the right of the Pause button. Choosing the abort button allows you to cancel an action that MicroPhone II would otherwise complete automatically. For instance, when you choose Wait for Call from the Phone menu, an Abort button appears, giving you the opportunity to discontinue the command.

The **Abort Script** button appears just as a script begins to run. Choose this button to terminate script execution. (However, Abort Script will not activate any log-off procedure even if the script succeeded in connecting to a service before you aborted.)

The **Abort Print** button appears whenever you choose a print command. This button will cancel the command and stop any additional data from being sent to the printer.

To interrupt a text transfer from your terminal to the remote computer, choose the **Abort Send** button. This button will appear when you send a text file from the Transfer menu or from a script.

The **Abort Paste** button appears during a paste action initiated from the Edit menu in the terminal window. Choosing this button will stop the paste.

Abort Script

Purpose: To stop the execution of a script.

Subcommands: None.

Syntax: Abort Script

Argument: None.

Result: MICROPHONE II aborts all currently running scripts.

Example:

```
1  Dial Service * "'MSEC"
2  If Failure
3        Abort Script
4  End If
5  Chain to Script "'Sign In'"
```

Alert *

Purpose: To display alerts on the screen.

The alert stops script execution and must be dismissed before execution can proceed.

Subcommands: Requires one of four subsidiary commands:

Alert * OK [ ]
Alert * OK/Abort [ ]
Alert * OK/Cancel [ ]
Alert * Yes/No [ ]

Alert * OK

Purpose:	To display an alert on the screen. The alert stops script execution and must be dismissed before execution can proceed.
Syntax:	Alert * OK [ ]
Argument:	String - enter the alert message in the expression box.
Result:	MICROPHONE II displays an alert with a button labeled OK.
Success flag:	This version of the command does not set the Success flag.
Example:	Alert * OK "'You have mail.'"

Alert * OK/Abort

Purpose: To display an alert on the screen.

The alert stops script execution and must be dismissed before execution can proceed.

Syntax: Alert * OK/Abort [ ]

Argument: String - enter the alert message in the expression box.

Result: MICROPHONE II displays an alert with a button labeled OK and another labeled Abort. Choosing the OK button allows script execution to continue; clicking the Abort button causes the script to abort.

Success flag: This version of the command does not set the Success flag.

Example: Alert * OK/Abort "'Unable to reach BB, keep trying?'"

Alert * OK/Cancel

Purpose:	To display an alert on the screen. The alert stops script execution and must be dismissed before execution can proceed.
Syntax:	Alert * OK/Cancel [ ]
Argument:	String - enter the alert message in the expression box.
Result:	MICROPHONE II displays an alert with a button labeled OK and another labeled Cancel.
Success flag:	Choosing the OK button returns True. Choosing the Cancel button returns False.
Example:	Alert * OK/Cancel "'Unable to reach ' & servicename & '--keep trying?'"

Alert * Yes/No

Purpose: To display an alert on the screen.

The alert stops script execution and must be dismissed before execution can proceed.

Syntax: Alert * Yes/No [ ]

Argument: String - enter the alert message in the expression box.

Result: MICROPHONE II displays an alert with a button labeled Yes and another labeled No.

Success flag: Choosing the Yes button returns True.
Choosing the No button returns False.

Example: Alert * Yes/No "'Retrieve your ' & count & ' pieces of mail?'"

Append to File *

Purpose: To append transmitted data to the end of a text file.

Subcommands: None.

Syntax: Append to File * [ ]

Argument: String - enter the name of the capture file in the expression box.

Result: MICROPHONE II opens the file and begins appending data as it arrives in the terminal window.

If no filename is entered into the expression box, you'll be prompted to name one during execution. If the file named in the expression box does not exist, MICROPHONE II will create it.

Once you've issued this command, use the Capture * Off script command to stop the capturing process and the Capture * On script command to continue, or use the equivalent File menu commands.

Success flag: Set to True if the capture file is successfully opened.
Set to False if the file cannot be opened.

Example: To add the date and time to the file, you might use the following script commands:

```
1  Append to File "'Capture.CIS'"
2  Set Variable * DT from Date & Time
3  Send Local to File   "'--------------------------------'"
4  Send Local to File   "'Added at '&DT"
```

Application *

Purpose: To start an application.

Subcommands: Requires one of two subsidiary commands:

Application * Load [ ]
Application * Run [ ]

Application * Load

Purpose: To start an application as an icon.

Syntax: Application * Load [ ]

Argument: String - enter the application's pathname

Result: MicroPhone II starts the application specified by the argument. The application will appear as an icon on the Windows desktop.

The pathname must be specified if the application is not on the path. Additional command line arguments may be included.

If no argument is entered into the expression box, you'll be prompted to name one during execution.

Success flag: Set to True if the application is successfully opened.
Set to False if the application cannot be opened.

Example: Application * Load "'MicPhone mci.mdc'"

Application * Run

Purpose: To start an application in a window.

Syntax: Application * Run [ ]

Argument: String - enter the application's pathname

Result: MicroPhone II starts the application specified by the argument. The application's window will appear on the screen. The new application receives the focus.

The full path must be specified if the application is not on the path. Additional command line arguments may be included.

If no argument is entered into the expression box, you'll be prompted to name one during execution.

Success flag: Set to True if the application is successfully opened.
Set to False if the application cannot be opened.

Example: Application * Run "'Excel'"

Beep

Purpose: To cause the computer to beep.

Subcommands: None.

Syntax: Beep

Argument: None.

Result: MicroPhone II gets your attention by beeping once.

Example:

```
1  Beep
2  Dialog Install "'You have urgent mail.'"
```

Capture *

Purpose: To start or stop saving the incoming text to a text file that has been specified through either the Open Capt. File * or Append to File * command.

Subcommands: Requires one of two subsidiary commands:

Capture * On
Capture * Off

Capture * On

Purpose: To start saving the incoming text to a text file that has been specified through either the Open Capt. File * or Append to File * command.

Syntax: Capture * On

Argument: None.

Result: MicroPhone II will begin saving all incoming and outgoing text into the disk file named in a preceding Open Capt. File * or Append to File * command.

Success flag: Set to True if the capture file is open.
Set to False if the capture file cannot be opened.

Example:
```
1  Wait for Text "'First Page'"
2  Capture * On
```

Capture * Off

Purpose: To stop saving the incoming text to a text file.

Syntax: Capture * Off

Argument: None.

Result: MicroPhone II halts the capture process. This closes the capture file, making it available to other programs.

Success flag: Set to True if the capture file is successfully closed.
Set to False if an error occurs when MicroPhone II tries to close the capture file.

Example:

```
1  Send Text String "'read mail^M'"
2  Open Capt. File * "'mymail'"
3  Wait For Text "'last page'"
4  Capture * Off
```

Chain To Script

Purpose: To pass execution to another script without returning to the next line of the calling script.

Subcommands: None.

Syntax: Chain to Script (script name)

Argument: Script - select from the subsidiary list box any script in the settings document.

Result: MicroPhone II passes script execution to the called script. Execution returns to the end of the calling script, skipping all script lines after the Chain to Script command. Variables retain their values.

Example:

```
1  When Text Equals "'You have mail.'"
2      Do Script * "'Process Mail'"
3  Or When Text Equals "'Mailbox empty.'"
4      Chain to Script "'Sign Off'"
5  End When
```

Note: If you change the name of the called script, Chain to Script will not find the renamed script. You must also change the script name in this script command.

Connection *

Purpose: To open, close and maintain a connection.

Subcommands: Requires one of five subsidiary commands:

Connection * Open
Connection * Close
Connection * Listen
Connection * Setup
Connection * Cleanup

Connection * Open

Purpose: To open a connection.

Syntax: Connection * Open

Argument: None.

Result: MicroPhone II opens the connection specified in the communications settings.

If a different connection is already open, it will be automatically closed and the new one opened.

Success flag: Set to True if the connection is successfully opened.
Set to False if the connection cannot be opened.

Example:
```
1  Connection * Open
2  Dial Service "'SVC BBS'"
```

Connection * Close

Purpose: To close a connection.

Syntax: Connection * Close

Argument: None.

Result: MicroPhone II closes the connection specified in the communications settings. This allows other applications to use the connection while MicroPhone II is running.

Success flag: Set to True if the connection is successfully closed.
Set to False if the connection cannot be closed.

Example: Connection * Close

Connection * Listen

Purpose: To check for characters received by a connection port.

Syntax: Connection * Listen

Argument: None.

Result: MicroPhone II listens to the connection port specified in the communications settings. Any characters that are received by the connection port are passed to the terminal window.

Success flag: Set to True if a listen connection is successfully established.
Set to False if a listen connection cannot be established.

Example:

```
1  Connection * Listen
2  Wait for Text "'RING' { the phone is ringing }"
3  Send Text String "'ATA^M' { answer it }"
```

Connection * Setup

Purpose: To set up a connection port.

Syntax: Connection * Setup

Argument: None.

Result: MicroPhone II executes a script named "Setup". If the script does not exist, MicroPhone II will return failure. Execution returns to the next script command at the completion of "Setup" (as in the Do Script command).

The script named "Setup" is a special script which is automatically executed when a communications driver is opened. See the Editing Communications Drivers chapter in the User's Guide for more information.

Success flag: Set to True if Setup is successfully executed.
Set to False if Setup cannot be executed.

Example: Connection * Setup

Connection * Cleanup

Purpose: To clean up a connection port.

Syntax: Connection * Cleanup

Argument: None.

Result: MicroPhone II executes the script named "Cleanup". If the script does not exist, MicroPhone II will return failure. Execution returns to the next script command at the completion of "Cleanup" (as in the Do Script command).

The script named "Cleanup" is a special script which is automatically executed when a communications driver is closed. See the Editing Communications Drivers chapter in the User's Guide for more information.

Success flag: Set to True if Cleanup is successfully executed.
Set to False if Cleanup cannot be executed.

Example:

```
1  Connection * Setup
2  Connection * Open
3  Dial Service * "'SVC BBS'"
4  Do Script "'Get my mail'"
5  Connection * Close
6  Connection * Cleanup
```

Cycle

Purpose: The Cycle command passes script execution immediately to the script statement containing the test condition within a While ... or a Repeat/Until loop.

Subcommands: None.

Syntax: Cycle

Argument: None.

Result: MicroPhone II passes script execution immediately to the previous While ... or the next Until ... command.

Example:

```
1   While Expression "count < maxtries"
2       Set Variable * count from Expression "count+1"
3       Send Text String "'ATDT'"
4       Send Text String "servicenumber"
5       Send Text String "'^M'"
6       When Line Contains "'BUSY'"
7           Cycle
8       Or When Line Contains "'ERROR'"
9           Alert*OK   "'Modem command is invalid.'"
10          Abort Script
11      Or When Line Contains "'NO CARRIER'"
12          Cycle
13      Or When Line Contains "'CONNECT'"
14          Abort Script
15      End When
16  End While
```

DDE *

Purpose: To coordinate DDE conversations between MicroPhone II, as the client, and other Windows applications, as servers. (No scripting is required for MicroPhone II to act as a server.)

Notes: Throughout these script commands, the variable name identifies the connection to the DDE server. It will be zero if the connection is not opened, non-zero if the connection is established. Do not change the value of this variable; it is used by other DDE commands to identify which server connection to use.

DDE messages are built from MicroPhone II expressions. See Script Language for a description of MicroPhone II's expressions, operators and functions.

For example, to include a single quotation mark in a DDE message, you must use two single quotes in the expression. Also, strings may be built by concatenating smaller strings together. The examples attached to each of the DDE subcommands further illustrate these rules.

Subcommands: Requires one of nine subsidiary commands:

DDE * Initiate (variable name) [ ]
DDE * Terminate (variable name)
DDE * Terminate All
DDE * Advise (variable name) [ ]
DDE * Unadvise (variable name) [ ]
DDE * Request (variable name) [ ]
DDE * Poke (variable name) [ ]
DDE * Execute (variable name) [ ]
DDE * Inhibit

DDE * Initiate

Purpose: To start a DDE conversation between MicroPhone II, as the client, and another Windows application, as server. (No scripting is required for MicroPhone II to act as a server.)

Syntax: DDE * Initiate (variable name) [ ]

Argument 1: String - in the expression box, enter the DDE name of the application with which you wish to open a DDE conversation.

Argument 2: String - in the expression box, enter a topic for the conversation.

Result: A conversation is opened with the DDE server specified by the first argument using the topic specified by the second argument. The variable name identifies the specific connection to the DDE server.

Success flag: Set to True if the connection is opened.
Set to False if no connection is established.

Example: DDE * Initiate DowJones2Excel "'Excel','System'"

See DDE *

DDE * Terminate

Purpose: To end a DDE conversation between MicroPhone II and another Windows application.

Syntax: DDE * Terminate (variable name)

Argument: None.

Result: Terminates the DDE conversation identified by the variable name. Any pending messages are discarded.

Success flag: Set to True if the connection is closed.
Set to False if the server doesn't respond. The connection is closed anyway.

Example: DDE * Terminate DowJones2Excel

See DDE *

DDE * Terminate All

Purpose: To end all DDE conversation that currently exists between MicroPhone II and other Windows applications.

Syntax: DDE * Terminate All

Argument: None.

Result: Terminates all DDE conversations. Any pending messages with any DDE servers are discarded.

Success flag: Set to True if the connections are closed.
Set to False if any server doesn't respond. The connections are closed anyway.

Example: DDE * Terminate All

See DDE *

DDE * Advise

Purpose: To create an automatically updated DDE link between MicroPhone II, as the client, and an item on another Windows application, as the server.

Syntax: DDE * Advise (variable name) [ ]

Argument 1: String - in the expression box, enter the name of the server item updated by the server. This string must be an item recognizable by the server application.

Argument 2: String - in the expression box, enter the name of the MicroPhone II variable to be updated when the server item updates.

Result: The variable named by the string in the second argument will be changed by the server whenever the corresponding item in the server application changes. If the variable does not exist, MicroPhone II will create it. This is known as a hot link.

Success flag: Set to True if the link is established.
Set to False if the server is unable to establish the link.

Example: DDE * Advise DowJones2Excel "'A3C12','Var1'"

See DDE *

DDE * Unadvise

Purpose: To drop the DDE link between MicroPhone II, as the client, and the server item of another Windows application, as the server.

Syntax: DDE * Unadvise (variable name) [ ]

Argument: String - in the expression box, enter the name of the server item updated by the server. This string must be an item recognizable by the server application. A null string will terminate all links.

Result: The link to the server item specified by the argument will be dropped.

Success flag: Set to True if the link is terminated.
Set to False if no such link exists.

Example: DDE * Unadvise DowJones2Excel "'A3C12'"

See DDE *

DDE * Request

Purpose: To create an update upon request DDE link between MicroPhone II, as the client, and another Window application, as the server.

Syntax: DDE * Request (variable name) [ ]

Argument 1: String - in the expression box, enter the name of the server item updated by the server. This string must be an item recognizable by the server application.

Argument 2: String - in the expression box, enter the name of the MicroPhone II variable to be updated with the server item.

Result: The variable named by the string in the second argument will be set to the current value of the server item. If the variable does not exist, it will be created. Unlike the Advise command, this update will only happen once.

Success flag: Set to True if the MicroPhone II variable is updated.
Set to False if the server is unable to satisfy the request.

Example: DDE * Request DowJones2Excel '"A3C12', 'Var1'"

See DDE *

DDE * Poke

Purpose: To update the server item of a Windows application, as the server, with a value from MicroPhone II, as the client.

Syntax: DDE * Poke (variable name) [ ]

Argument 1: String - in the expression box, enter the name of the server item updated by the server. This string must be an item recognizable by the server application.

Argument 2: String - in the expression box, enter the value to be passed to the server item.

Result: The server item specified in the first argument will be set to the value of the second argument.

Success flag: Set to True if the server item is updated.
Set to False if the server is unable to process the item.

Example:

```
1  Set Variable * adate from Expression "'Today''s date is ' & month '/' & day & '/' & year
"
2  DDE * Poke DowJones2Excel "'A3', adate"
```

See DDE *

DDE * Execute

Purpose: To send a command to another Windows application linked as a server to MicroPhone II.

Syntax: DDE * Execute (variable name) [ ]

Argument: String - in the expression box, enter the command, or string of commands, to be executed by the server.

Result: The server is directed to execute the commands within the square brackets.

Success flag: Set to True if the server accepts the commands.
Set to False if the server is unable to process the commands.

Example: DDE * Execute DowJones2Excel '"[OPEN("WORKSHT.XLS")]"'

See DDE *

DDE * Inhibit

Purpose: To disable processing of DDE messages addressed to MicroPhone II.

Syntax 9: DDE * Inhibit

Options: No and Yes - select one from the third list box.

Argument: None.

Result: If Yes is chosen, MicroPhone II will ignore all DDE messages addressed to the settings document that invokes this command. All currently open DDE conversations will be terminated automatically. This setting is not saved in the settings document; it must be set from scripting each time the document is opened.

If No is chosen, MicroPhone II will allow DDE conversations. No is the default.

Success flag: This version of the command does not set the Success flag.

Example:

```
1  DDE * Inhibit Yes
2  Send File * ZMODEM Batch "'mybatch.mbt'"
3  DDE * Inhibit No
```

See DDE *

Delete

Purpose: To remove a variable from MicroPhone II's list of variables, or to remove all variables at once. Variables are created when a value is assigned to them and remain available until you delete them or quit the program.

Subcommands: Requires one of two subsidiary commands:

Delete Variable (variable name)
Delete All Variables

Delete Variable

Purpose: To remove a variable from MicroPhone II's list of variables. Variables are created when a value is assigned to them and remain available until you delete them or quit the program.

Syntax: Delete Variable (variable name)

Argument: Enter the name of a variable in the Variable text box.

Result: This command removes the variable named in the Variable box.

If the variable does not exist, script execution simply passes to the next statement.

Example:

```
1  Set Variable * adate from Expression '"Today"s date is ' & month & '/' & day & '/' & year "
2  Alert * OK "adate"
3  Delete Variable adate
```

Delete All Variables

Purpose: To remove all variables at once from MicroPhone II's list of variables. Variables are created when a value is assigned to them and remain available until you delete them or quit the program.

Syntax: Delete All Variables

Argument: None.

Result: This command removes all of the variables created since the program was started.

Example:

```
1   If Expression "initialize = true"
2       Delete All Variables
3   Else
4       Do Script * "'Save Variables'"
5   End If
```

Dial Service *

Purpose: To dial a phone number that was previously entered in the Create Service dialog box.

Subcommands: None.

Syntax: Dial Service * (service name)

Argument: Service - select a service from the subsidiary list box.

Result: MicroPhone II gets the modem's attention, dials the phone number, waits for a connection, and sets the Success flag accordingly.

Success flag: Set to True if the call gets through.
Set to False if no connection is established.

Example:

```
1  Repeat
2      Dial Service * "'CIS'"
3  Until Success
```

Dialog

Purpose: To place a dialog box on the screen that contains information about the action currently in progress.

A dialog box installed via the Dialog É command does not stop execution of scripts.

Subcommands: Requires one of three subsidiary commands:

Dialog Install [ ]
Dialog Update [ ]
Dialog Discard

Dialog Install

Purpose: To place a dialog box on the screen that contains information about the action currently in progress.

A dialog box installed via the Dialog command does not stop execution of scripts.

Syntax: Dialog Install [ ]

Argument: String - in the expression box, enter a prompt to appear in the dialog box.

Result: Causes a dialog box to appear. The dialog box will remain on the screen until the command Dialog Discard is executed or until MicroPhone II ceases executing scripts and returns to terminal mode.

If a dialog box is already present and the script attempts to install another, the new text string will replace the old and the dialog will remain on the screen.

Example: Dialog Install "'Checking for mail ...'"

Dialog Update

Purpose: To change the message on a dialog box currently on the screen.

Syntax: Dialog Update [ ]

Argument: String - in the expression box, enter a prompt to appear in the dialog box.

Result: MicroPhone II changes the message in the dialog so that it displays the prompt that was entered in the expression box. If a dialog box has not yet been installed, this command will do so.

Example: Dialog Update "'You have ' & count & ' pieces of mail.'"

Dialog Discard

Purpose: To remove a dialog box from the screen.

Syntax: Dialog Discard

Argument: None.

Result: Removes the dialog box from the screen. Dialog boxes installed by the Dialog Install and Dialog Update commands can be dismissed with this command.

Example: Dialog Discard

Do Script *

Purpose: To execute another script and then return to the current one.

Subcommands: None.

Syntax: Do Script * (script name)

Argument: Script - select from the subsidiary list box any script in the settings document.

Result: The calling script is temporarily halted, and the called script is executed until a Return command is encountered or the end of the script is reached. The original script will then continue executing at the statement following the Do Script * command. Variables retain their values as execution passes from one script to another.

Success flag: Set to True if the Return command returns Success.
Set to False if the Return command returns Failure.

If the end of the script is reached without finding a Return command (or a command that sets the flag), the value of the flag is unchanged.

Example:

```
1   Do Script * "'Compose'"
2   If Success
3       Set Variable * mail2go from Expression "True"
4       Chain to Script "'Choices'"
5   End If
```

Edit

Purpose: To edit selected text in the terminal window and the buffer. The operation of the Edit commands is identical to that of the commands in the Edit menu.

Subcommands: Requires one of seven subsidiary commands:

Edit Cut
Edit Copy
Edit Paste
Edit Clear
Edit Copy Table
Edit Select All
Edit Clear Buffer

Edit Cut

Purpose: To remove selected text.

Syntax: Edit Cut

Argument: None.

Result: MicroPhone II removes selected text, and places it on the Clipboard.

Example:

```
1  Selection * Begin
2  Move Cursor to Absolute Row "24"
3  Move Cursor to Absolute Column "80"
4  Election * Extend
5  Edit Cut
```

Edit Copy

Purpose: To copy selected text.

Syntax: Edit Copy

Argument: None.

Result: MicroPhone II duplicates selected text and places the copy on the Clipboard.

Example:

```
1  Selection * Begin
2  Move Cursor to Absolute Row "1"
3  Move Cursor to Absolute Column "1"
4  Selection * Extend
5  Edit Copy
```

Edit Paste

Purpose: To place text from the Clipboard into MicroPhone II.

Syntax: Edit Paste

Argument: None.

Result: MicroPhone II pastes data from the Clipboard into the terminal window, thus sending it to the remote computer.

Example: To assign two actions to one F-Key, set a variable such as FKeySwitch to identify the action and assign an F-Key to this script. Invoking the F-Key tests the value of the variable to determine whether to send the contents of the Clipboard or to send the contents of another variable, signature.

```
1   If Expression "FKeySwitch = False"
2       Edit Paste
3   Else
4       Send Text String "signature & '^M'"
5   End If
```

Edit Clear

Purpose: To remove selected text without placing it in the Clipboard.

Syntax: Edit Clear

Argument: None.

Result: MicroPhone II deletes selected text but does not place it in the Clipboard.

Example:
```
1  Edit Select All
2  Edit Clear
```

Edit Copy Table

Purpose: To copy selected text in a tabular format.

Syntax: Edit Copy Table

Argument: None.

Result: MicroPhone II copies selected data to the Clipboard. The command transforms any occurrence of two or more spaces in the selected data to a Tab character (the Copy command leaves them as spaces). Existing screen tabs are copied as is.

Example:

```
1  Selection * Begin
2  Move Cursor to Relative Row "-23"
3  Move Cursor to Relative Column "-79"
4  Selection * Extend
5  Edit Copy Table
```

Edit Select All

Purpose: To select all the text in MicroPhone II's buffer and terminal window.

Syntax: Edit Select All

Argument: None.

Result: MicroPhone II selects all the data in the buffer and the terminal window.

Example:

```
1  Edit Select All
2  Selection * Append "'Session History'"
```

Edit Clear Buffer

Purpose: To remove all the text from the screen buffer.

Syntax: Edit Clear Buffer

Argument: None.

Result: MicroPhone II clears all data from the screen buffer.

Example:

```
1   If Expression "logoff"
2       Edit Clear Buffer
3       Chain to Script "'Log Off'"
4   End If
```

Else

Purpose: To serve as an optional clause in an If construct. The program will execute all statements in the Else clause if no other condition in the If construct is met.

Subcommands: None.

Syntax: Else

Argument: None.

Result: MicroPhone II executes the statements following the Else.

Example:

```
1   If Expression "counter < mailcount"
2       Do Script * "'Get Mail'"
3   Else If Expression "counter >= mailcount"
4       Dialog Install "'All mail received.'"
5   Else
6       Alert * OK "'Error - problem receiving mail'"
7   End If
```

Else If

Purpose: To introduce an alternate condition upon which script execution may be passed to the statement following the If construct.

The Else If command must be used as part of an If construct. The If construct can contain multiple Else If statements.

Subcommands: Requires one of six subsidiary commands:

These subcommands function identically to the corresponding If subsidiary commands.

Else If Text Equals [ ]
Else If Line Contains [ ]
Else If Line Does Not Contain [ ]
Else If Failure
Else If Success
Else If Expression [ ]

Example:

```
1   If Expression "mail2go"
2       Do Script * "'Send Mail'"
3   Else If Expression "mailhold"
4       Do Script * "'Prepare Mail'"
5   Else
6       Return Success
7   End If
```

End If

Purpose: To close an If construct.

Subcommands: None.

Syntax: End If

Argument: None.

Result: MicroPhone II passes script execution to the statement following the If construct.

Example:

```
1   Do Script * "'Initialize Parameters'"
2   If Success
3       Dial Service * "servicename"
4   End If
```

End When

Purpose: To close a When construct.

Subcommands: None.

Syntax: End When

Argument: None.

Result: If none of the conditions in the When construct is met, MicroPhone II passes script execution to the top of the construct. If one of the conditions in the When construct is met, MicroPhone II executes the block of statements following that condition and then passes execution to the statement following the End When command.

Example:

```
1   When Text Equals "'NO CARRIER'"
2       Do Script * "'Reset Board'"
3   Or When Seconds Have Passed "120"
4       Do Script * "'Reset Board'"
5   End When
```

End While

Purpose: To close a While construct.

Subcommands: None.

Syntax: End While

Argument: None.

Result: If the condition in the While construct is satisfied, MicroPhone II will pass script execution to the top of the construct. If the condition in the While construct is not satisfied, MicroPhone II passes script execution to the statement following the End While command.

Example:

```
1   While Failure
2       Dial Service * "servicename"
3   End While
```

File *

Purpose: To access any file on any available volume for input, output or file management.

Before issuing File * ... with the Close, Close All, Write, Set Position or Set EOF subcommands, you must first open the file with the File * Open command. On the other hand, Delete, Copy, Rename and Move may only be performed on files that are not open.

A filemark in an open file designates the point at which an operation starts.

Notes: Do not apply the File * ... commands to files that have been opened with other commands, such as Open Capt. File * ... or Send File * Text Line by Line.

If a file has been opened and then closed by these commands, you may subsequently apply a File * ... command to it.

A number of functions are particularly useful with the File * ... script command: EOF, EOLN, FileLen, FilePos, GetEOF, ReadCh, ReadLn and ReadStr. (See Functions for more information.)

Subcommands: Requires one of eleven subsidiary commands:

File * Open [ ]
File * Close [ ]
File * Close All
File * Write [ ]
File * Set Position at Offset [ ]
File * Set Position at Line [ ]
File * Set EOF [ ]
File * Delete [ ]
File * Copy [ ]
File * Rename [ ]
File * Move [ ]

File * Open

Purpose: To open any file on any available volume for input, output or file management.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Open [ ]

Argument: String - in the expression box, enter the name of the file to open.

Result: MicroPhone II opens the file and sets the filemark to zero. If the named file cannot be found, MicroPhone II will create a file with the name and path you provided in the argument.

Success flag: Set to True if the file is successfully opened.
Set to False if the file cannot be opened.

Example:
```
1  Set Variable * inbox from Expression "'c:\mp\svc\mail\MailFile.txt'"
2  File * Open "inbox"
```

File * Close

Purpose: To close a file.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Close [ ]

Argument: String - in the expression box, enter the name of the file to close.

Result: MicroPhone II closes a file that you have previously opened with the File * Open command.

Success flag: Set to True if the file is successfully closed.
Set to False if the file is not open or cannot be closed.

Example:

```
1  Do Script * "'Sweep Mail'"
2  If Failure
3        File * Close "inbox"
4  End If
```

File * Close All

Purpose: To close all currently open files.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Close All

Argument: None.

Result: MicroPhone II closes all currently open files that have been opened with the File * Open command.

Success flag: Set to True if all open files are successfully closed.
Set to False if there are no open files or if any of the open files cannot be closed.

Example:

```
1   If Expression "MONTH > currentMonth"
2       File * Close All
3   End If
```

File * Write

Purpose: To write data to an open file.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Write [ ]

Argument 1: String - in the expression box, enter the name of the file.

Argument 2: String - following the filename (Argument 1), type a comma and then enter the data to write to the file.

Result: MicroPhone II writes the data from Argument 2 in the designated file, starting from the filemark. The filemark will be advanced to just beyond the end of what is written.

If the filemark is located before the end of the file when the File * Write command is invoked, any data in the file after the filemark will be overwritten and lost.

The file must be open before this command is issued.

Success flag: Set to True if the data is written.
Set to False if the data cannot be written.

Example:

```
1  File * Open "myFile"
2  File * Open "mailFile"
3  Set Variable * X From Expression "ReadLn(mailFile)"
4  File * Write "myFile,X"
```

File * Set Position at Offset

Purpose: To set the filemark in an open file.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Set Position at Offset [ ]

Argument 1: String - in the expression box, enter the name of the file.

Argument 2: Integer - following the filename (Argument 1), type a comma and then enter the number of bytes from the first character in the file (offset zero).

Result: MicroPhone II positions the filemark in the file. If you attempt to position the filemark at an offset that is beyond the end of the file, the filemark will be set to the end of the file instead.

The file must be open before this command is issued.

Success flag: Set to True if the position is set successfully.
Set to False if the position cannot be set.

Example:

```
1   File * Set Position at Offset "incomingFile, 25"
2   Set Variable * testchar from Expression "ReadCh (incomingFile)"
3   If Expression "testchar = Char(27)"
4       Do Script * "'ConvertFile'"
5   End If
```

File * Set Position at Line

Purpose: To set the filemark to the beginning of a line in an open file.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Set Position at Line [ ]

Argument 1: String - in the expression box, enter the name of the file.

Argument 2: Integer - following the filename (Argument 1), type a comma and then enter the number of lines from the first line in the file (line zero).

Result: MicroPhone II positions the filemark at the beginning of the line specified by Argument 2. If you attempt to position the filemark at a line that is beyond the end of the file, the filemark will be set to the end of the file instead.

The file must be open before this command is issued.

Success flag: Set to True if the position is set successfully.
Set to False if the position cannot be set.

Example:

```
1  File * Open "myFile"
2  File * Set Position at Line "myFile,10"
3  Set Variable * teststring from Expression "ReadLn(myFile)"
```

File * Set EOF

Purpose: To set the end of file in an open file.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Set EOF [ ]

Argument 1: String - in the expression box, enter the name of the file.

Argument 2: Integer - following the filename (Argument 1), type a comma and then enter the number of characters from the beginning of the file where the EOF mark should be set.

Result: MicroPhone II moves the end of the file to the offset specified. If you truncate the file, data may be lost. If you lengthen the file, random data will occupy the new portion of the file until valid data is written to it.

The file must be open before this command is issued.

Success flag: Set to True if the end of the file is set successfully.
Set to False if the end of the file cannot be set.

Example: To set the end of file after character 499:

```
1  File * Open "myFile"
2  File * Set EOF "myFile,500"
```

File * Delete

Purpose: To delete a file.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Delete [ ]

Argument: String - in the expression box, enter the name of the file to delete. (This command takes file names only; it does not delete directories.)

Result: MicroPhone II removes the file from disk. The file must be closed before this command is issued.

Success flag: Set to True if MicroPhone II successfully deletes the file.
Set to False if the file cannot be deleted.

Example:
```
1  File * Close "oldMail"
2  File * Delete "oldMail"
```

File * Copy

Purpose: To copy a file.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Copy [ ]

Argument 1: String - in the expression box, enter the name of the file to copy. (This command takes filenames only; it does not copy directories.)

Argument 2: String - following the filename (Argument 1), type a comma and then enter the name of the destination directory where the duplicate should reside. If you also want to change the file's name, type the new name after the destination directory.

Result: MicroPhone II copies the file and places the duplicate in the designated directory. If a filename was included in Argument 2, the copy will be named accordingly. The file must be closed before this command is issued.

Success flag: Set to True if MicroPhone II is able to copy the file.
Set to False if the file cannot be copied.

Example:
```
1  File * Close "mailfile"
2  File * Copy "'c:\mp\svcbbs\mail\mailfile', 'Mail' & month & day"
```

File * Rename

Purpose: To change the name of a file.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Rename [ ]

Argument 1: String - in the expression box, enter the old filename of the file. The pathname must be included if the file is not in the current directory. (This command takes file names only; it does not rename directories.)

Argument 2: String - following the old filename (Argument 1), type a comma and then enter the new name of the file. Do not include a pathname; this command is only for renaming and not for moving.

Result: MicroPhone II changes the name of the file in Argument 1 to the name in Argument 2. The file must be closed before this command is invoked.

Success flag: Set to True if the file is successfully renamed.
Set to False if the file cannot be renamed.

Example:

```
1  Set Variable * mailFile from Expression "'SettingsPath & newmail.txt'"
2  Set Variable * oldMail from Expression "month & day & '.txt'"
3  File * Close "mailFile"
4  File * Rename "mailFile, oldMail"
```

File* Move

Purpose: To move a file.

See File * ... for a more detailed explanation on when and how to use the File * ... commands.

Syntax: File * Move [ ]

Argument 1: String - in the expression box, enter the filename of the file to move. (This command moves files only; it will not move directories.)

Argument 2: String - following the filename (Argument 1), type a comma and then enter the pathname of the directory where the file should be moved. Do not include a filename; this command is only for moving and not for renaming.

Result: MicroPhone II moves the file to the designated directory. The file must be closed before this command is invoked.

Success flag: Set to True if the file is successfully moved.
Set to False if the file cannot be moved.

Example:

```
1  File * Close "'C:capture\mailFile.txt'"
2  File * Move "'C:capture\mailFile.txt', 'C:oldmail'"
```

Hang Up

Purpose: To break an established connection.

Subcommands: None.

Syntax: Hang Up

Argument: None.

Result: MicroPhone II instructs the modem to disconnect using the modem script "Hang Up."

Example:

```
1   Do Script * "'Check Mail'"
2   If Expression "Mailcount = 0"
3       Do Script * "'Sign Off'"
4       Hang Up
5   End If
```

If

Purpose: To execute a block of statements as soon as any of a number of conditions is met.

The If construct is "static" in the sense that it does not move characters from the serial port to the terminal window. It should not be used to check for test strings as data is being received, but only when data transmission is halted. To test for new data as it streams into the terminal window, use the Wait ... or When ... command.

If is a logical construct; See Logical Constructs for more information.

The tests in the If construct are tried in order, from If ... to the last Else If ... command. When one of the tests is evaluated as True, the script first executes the block of statements following that test and then the statements following the End If command.

If none of the tests are evaluated as True, then the script executes the block of statements following the Else statement, provided the construct includes one.

If the construct does not contain an Else clause, and none of the If conditions are True, then no block of statements within the construct will be executed. Script execution will then pass to the statement following the End If command.

An If construct may contain multiple Else If ... commands in but at most one Else command. Every If construct must end with an End If command.

Subcommands: Requires one of six subsidiary commands:

If Text Equals [ ]
If Line Contains [ ]
If Line Does Not Contain [ ]
If Failure
If Success
If Expression [ ]

If Text Equals

Purpose: To execute a block of statements if the result of a text string comparison is True.

See If for a more detailed explanation of the If construct.

Syntax: If Text Equals [ ]

Argument: String - enter a test string in the expression box.

Result: This test will be considered True if the text immediately to the left of the cursor on the current line is the same as the test string that is entered into the expression box.

Example:

```
1   Wait for Text "'CE'"
2   If Text Equals "'NOTICE'"
3       Printer On
4   End If
```

If Line Contains

Purpose: To execute a block of statements if the line contains the specified text.

See If for a more detailed explanation of the If construct.

Syntax: If Line Contains [ ]

Argument: String - enter a test string in the expression box.

Result: This test is considered True if the line above the cursor contains the test string entered into the expression box.

This test is valid when the cursor is located at the far left position on a line - at column one - such as following a carriage return. Otherwise, the test will return False.

Example:

```
1   Wait for Line Containing "'C'"
2   If Line Contains "'NO CARRIER'"
3       Return Failure
4   End If
```

If Line Does Not Contain

Purpose: To execute a block of statements if the line does not contain the specified text..

See If for a more detailed explanation of the If construct.

Syntax: If Line Does Not Contain [ ]

Argument: String - enter a test string in the expression box.

Result: This test is considered True if the line above the cursor does not contain the test string entered into the expression box.

This test is valid when the cursor is located at the far left position on a line - at column one - such as following a carriage return. Otherwise, the test will return False.

Example:

```
1   Wait for Line Containing "'C'"
2   If Line Does Not Contain "'CONNECT'"
3       Return Failure
4   End If
```

If Failure

Purpose: To execute a block of statements if the Success flag is False.

See If for a more detailed explanation of the If construct.

Syntax: If Failure

Argument: None.

Result: This test will be considered True if the Success flag is evaluated as False.

Example: If the connection with CIS is not established, the If test will be considered True:

```
1    Dial Service * "'CIS'"
2    If Failure
3        Do Script * "'MCI'"
4    End If
```

If Success

Purpose: To execute a block of statements if the Success flag is True.

See If for a more detailed explanation of the If construct.

Syntax: If Success

Argument: None.

Result: This test will be considered True if the Success flag is evaluated as True.

Example:

```
1   Send File * XMODEM Plain "'Project1.doc'"
2   If Success
3       Alert * Yes/No "'Send Second File?'"
4       If Success
5           Send File * XMODEM Plain "'Project2.doc'"
6       End If
7   Else
8       Alert * OK   "'The first file was not sent.'"
9   End If
```

If Expression

Purpose: To execute a block of statements if Expression is True.

See If for a more detailed explanation of the If construct.

Syntax 6: If Expression [ ]

Argument: Boolean - enter an expression that evaluates to True or False in the expression box.

Result: This test will be evaluated as True if the expression entered into the expression box is evaluated as True.

Example:

```
1   If Expression "DayOfWeek > 1 AND DayOfWeek < 7"
2       Do Script * "'Daily Mail'"
3   End If
```

Install Button

Purpose: To add a button to the button bar.

Subcommands: None.

Syntax: Install Button (script name)

Argument: Script - select from the subsidiary list box any script in the settings document.

Result: After the script containing this statement is finished running, MicroPhone II installs the button to the right of those previously installed.

The button remains on the bar until you remove it or change a button's status in an existing script. If you close the settings document and re-open it, the button will be absent until you re-install it.

Example:

```
1  Remove All Buttons
2  Install Button "'Compose'"
3  Install Button "'Log Off'"
4  Install Button "'Return'"
```

Leave

Purpose: To exit a While ... or Repeat/Until ... loop.

Subcommands: None.

Syntax: Leave

Argument: None.

Result: MicroPhone II passes script execution to the statement immediately after the next End While or Until ... command.

Example:

```
1   While Expression "count < maxtries"
2       Set Variable * count from Expression "count+1"
3       Send Text String "'ATDT'"
4       Send Text String "servicenumber"
5       Send Text String "'^M'"
6       When Line Contains "'CONNECT'"
7           Leave
8       Or When Line Contains "'BUSY'"
9           Cycle
10      Or When Line Contains "'NO CARRIER'"
11          Abort Script
12      End When
13  End While
```

Message Box *

Purpose: To open a Message Box for creating, reading, editing and replying to messages stored in text files. All of the standard editing commands are available in the Message Box.

Subcommands: Requires one of four subsidiary commands:

Message Box * Create File [ ]
Message Box * Edit File [ ]
Message Box * Read File [ ]
Message Box * Read File and Reply [ ]

Message Box * Create File

Purpose:	To open a Message Box for creating a text file where messages will be stored. All of the standard editing commands are available in the Message Box.
Syntax:	Message Box * Create File [ ]
Argument 1:	String - in the expression box, enter a title for the Message Box.
Argument 2:	String - in the expression box, enter a filename for the text file that you will create.
Result:	MicroPhone II displays the Message Box, where you may enter a message. Choose Exit (from the File menu) to save the message and exit.
Success flag:	Set to True if the file is created successfully. Set to False if the file is not created.
Example:	Message Box * Create File "'Enter Message', 'OutBox.txt'"

Message Box * Edit File

Purpose: To open a Message Box for editing the text file where messages are stored. All of the standard editing commands are available in the Message Box.

Syntax: Message Box * Edit File [ ]

Argument 1: String - in the expression box, enter a title for the Message Box.

Argument 2: String - in the expression box, enter the name of the text file that you want to edit.

Result: MicroPhone II displays the Message Box with the contents of the file specified in the second argument. Choose Exit (from the File menu) to save the message and exit.

Success flag: Set to True if the changes are successfully written to the file.
Set to False if the changes are not written to the file.

Example:
```
1  Set Variable * textfile from Text File Dialog
2  Message Box * Edit File "'Editing ' & textfile, textfile"
```

Message Box * Read File

Purpose: To open a Message Box for reading messages stored in a text file.

Syntax: Message Box * Read File [ ]

Argument 1: String - in the expression box, enter a title for the Message Box.

Argument 2: String - in the expression box, enter the name of the text file that you want to read.

Result: MicroPhone II displays the Message Box with the contents of the file specified in the second argument. Choose Exit (from the File menu) after reading the file. (Note that this command does not allow you to edit the file.)

Success flag: Set to True if the file is opened successfully.
Set to False if the file cannot be opened.

Example:

```
1  If Expression "help = 5"
2  Message Box * Read File "'Search Strategies', 'Dialog.txt'"
3  End If
```

Message Box * Read File and Reply

Purpose: To open a Message Box for reading the text file where messages are stored and replying to those messages. All of the standard editing commands are available in the Message Box.

Syntax: Message Box * Read File and Reply [ ]

Argument 1: String - in the expression box, enter a title for the Message Box.

Argument 2: String - in the expression box, enter the name of the text file that you want to read.

Argument 3: String - in the expression box, enter a file name for the text file where you want to save your reply. If the file you name does not exist, MicroPhone II will create it.

Result: MicroPhone II displays the Message Box with the contents of the file specified in the second argument. You should select Reply (from the File menu), type the reply. Then, choose Exit (from the File menu) to save the message and exit.

Success flag: Set to True if a reply file is successfully created, and it is not empty.
Set to False if the reply file is empty, or if it cannot be created.

Example: Message Box * Read File and Reply "'Enter Reply', 'InBox.txt', 'OutBox.txt'"

Move Cursor To

Purpose: To move the cursor to a specified position in the terminal window. This command affects only the local screen display. It does not send command sequences to a host system, and therefore, will not change the text insertion point.

The terminal window always occupies the bottom 24 lines of the session data, and either 80 or 132 columns across. The cursor will not move beyond the terminal window. If you enter a number larger than 24 x 80, or 24 x 132, the cursor will stop at the edge of the window.

Note: You may use Move Cursor to ... to customize your screen display. In particular, you can use the function NextCh to watch for incoming characters before they are displayed on the screen, convert them at will, and then draw the screen to your specifications with the use of screen addressing and Send Local to Screen script commands.

Subcommands: Requires one of four subsidiary commands:

Move Cursor to Absolute Row [ ]
Move Cursor to Absolute Column [ ]
Move Cursor to Relative Row [ ]
Move Cursor to Relative Column [ ]

Move Cursor To Absolute Row

Purpose: To move the cursor to a specified row in the terminal window.

See Move Cursor To for more information.

Syntax: Move Cursor to Absolute Row [ ]

Argument: Integer - in the expression box, enter the row position from 1 to the Rows setting in the Terminal Settings dialog box.

Result: MicroPhone II defines an absolute position for the cursor when the extreme upper left position of the terminal window is defined as row 1, column 1.

Example: To move the cursor to row 24, use the command:

Move Cursor to Absolute Row "24"

Move Cursor To Absolute Column

Purpose: To move the cursor to a specified column in the terminal window.

See Move Cursor To for more information.

Syntax: Move Cursor to Absolute Column [ ]

Argument: Integer - in the expression box, enter the column position from 1 to the Columns setting in the Terminal Settings dialog box.

Result: MicroPhone II defines an absolute position for the cursor when the extreme upper left position of the terminal window is defined as row 1, column 1.

Example: To move the cursor to column 3 of the current line, use the command:

Move Cursor to Absolute Column "3"

Move Cursor To Relative Row

Purpose: To move the cursor to a row in the terminal window relative to its current position.

See Move Cursor To for more information.

Syntax: Move Cursor to Relative Row [ ]

Argument: Integer - in the expression box, enter the change in row position. This value must be within the bounds set in the Terminal Settings dialog box.

Result: MicroPhone II defines a new position for the cursor relative to its current position.

Example: To move the cursor 2 spaces up, use the command:
Move Cursor to Relative Row "-2"

Move Cursor To Relative Column

Purpose: To move the cursor to a column in the terminal window relative to its current position.

See Move Cursor To for more information.

Syntax: Move Cursor to Relative Column [ ]

Argument: Integer - in the expression box, enter the change in column position. This value must be within the bounds set in the Terminal Settings dialog box.

Result: MicroPhone II defines a new position for the cursor relative to its current position.

Example: To move the cursor 3 spaces to the right, use the command:

Move Cursor to Relative Column "3"

Notify

Purpose: To post events when MicroPhone II is inactive.

Subcommands: Requires one of two subsidiary commands:

Notify Install [ ]
Notify Discard

Notify Install

Purpose: To activate notification when MicroPhone II is inactive.

Syntax: Notify Install [ ]

Argument: String - in the expression box, enter an optional message.

Result: A message appears on-screen if a string has been entered in the expression box. The system will beep and the settings document's title bar will flash. If the document is minimized, the icon will flash continuously.

If a message is shown, script execution will pause until you click its OK button. Clicking the button will dismiss the message and allow script execution to continue.

If no prompt is entered in the expression box, the message will not appear, but the flashing title bar will be activated or the icon will flash if the document is minimized. The system will beep when the command is executed.

To deactivate, issue the Notify Discard command.

Example:

```
1  If Expression "mail2go"
2      Notify Install "'You have mail.'"
3      Notify Discard
4  End If
```

Notify Discard

Purpose: To deactivate notification when MicroPhone II is inactive.

Syntax: Notify Discard

Argument: None.

Result: MicroPhone II removes the flashing element of the notification.

Example:

```
1  When Button "'Get Mail'"
2  Or When Seconds Have Passed "360"
3  End When
4  Notify Discard
```

Open Capture File

Purpose: To open a file on disk where transmitted data is to be saved.

Subcommands: None.

Syntax: Open Capt. File * [ ]

Argument: String - in the expression box, enter the name of the capture file.

Result: MicroPhone II automatically opens the file for capture; no Capture * On command is required.

If this file already exists on the disk, new data you capture will overwrite the file's contents. If no file name is entered into the expression box, you'll be prompted for the filename on script execution.

Success flag: Set to True if the file is successfully opened.
Set to False if the file cannot be opened.

Example: Open Capt. File * "'MSG' & Month & '-' & Day & '.TXT'"

Or When

Purpose: To introduce another condition or test in a When ... construct which, if evaluated as True, causes the script to execute the block of statements following the test, and then to pass execution to the statement following the End When command.

The Or When ... command must be used as part of a When ... construct. A construct may contain multiple Or When ... statements.

Subcommands: Requires one of eight subsidiary commands:

These commands function identically to the corresponding When ... subsidiary commands.

Or When Text Equals [ ]
Or When Line Contains [ ]
Or When Line Does Not Contain [ ]
Or When Time Is [ ]
Or When Seconds Have Passed [ ]
Or When Sixtieths Have Passed [ ]
Or When Button [ ]
Or When Expression [ ]

Example:

```
1   When Text Equals "'password'"
2       Do Script * "'Sign In'"
3   Or When Line Contains "'NO CARRIER'"
4       Chain to Script "'Log On'"
5   Or When Time Is "'17:30'"
6       Chain to Script "'Shut Down'"
7   Or When Button "'Cancel'"
8       Chain to Script "'Master Control'"
9   Or When Seconds Have Passed "300"
10      Chain to Script "'Log On'"
11  End When
```

Printer

Purpose: To toggle between sending and not sending transmitted data to the printer. This command controls print streaming, which is also known as print logging or real time printing.

Subcommands: Requires one of two subsidiary commands:

Printer On
Printer Off

Printer On

Purpose: To begin sending transmitted data to the printer.

Syntax: Printer On

Argument: None.

Result: MicroPhone II sends all text that appears in the terminal window to the printer. The printer must be on-line. Text is printed in the printer's text mode.

Example:

```
1  Wait for Text "'messages'"
2  Printer On
```

Printer Off

Purpose: To stop sending transmitted data to the printer.

Syntax: Printer Off

Argument: None.

Result: MicroPhone II stops echoing text to the printer.

Example:

```
1   When Text Equals "'end"'
2       Printer Off
3   End When
```

Quit MicroPhone

Purpose To stop script execution and return to Windows File Manager.

Subcommands: None.

Syntax: Quit MicroPhone

Argument: None.

Receive File *

Purpose: To receive files using any of the transfer protocols supported in MicroPhone II.

File transfer settings which are in effect at the time of the transfer apply. The receive command overrides the transfer mode setting for the duration of the transfer.

Files will be received into the directory specified by the Select Receive Dir ... script command, or from the Transfer menu.

MicroPhone II accepts a full pathname so you can receive files into a specific directory. However, the filename, or pathname, you supply will be ignored if a filename is received as part of the transfer.

If for any reason the program cannot find a filename for the received file, it will give the file a temporary filename (e.g., MIC00001.TMP). This applies to all of the program's protocols.

Note: The "Plain" designation in the Receive File * ... subcommands is included for cross-platform compatibility. It distinguishes from the MacBinary file transfer option.

Subcommands: Requires one of four subsidiary commands:

Receive File * XMODEM Plain [ ]
Receive File * YMODEM Plain [ ]
Receive File * YMODEM-G Plain [ ]
Receive File * ZMODEM Plain [ ]

Receive File * XMODEM Plain

Purpose: To receive files using the XMODEM protocol.

See Receive File * for more information.

Syntax: Receive File * XMODEM Plain [ ]

Argument: String - in the expression box, enter the name of the file.

Result: MicroPhone II will receive the file using the XMODEM protocol.

Success flag: Set to True if the file is received successfully.
Set to False if the transfer fails or is canceled.

Example: Receive File * XMODEM Plain "'FALLBACK.TXT'"

Receive File * YMODEM Plain

Purpose: To receive files using the YMODEM protocol.

See Receive File * for more information.

Syntax: Receive File * YMODEM Plain [ ]

Argument: String - in the expression box, enter the name of the file (optional).

Result: MicroPhone II will receive a single file or a batch of files using the YMODEM protocol.

Success flag: Set to True if all files are received successfully.
Set to False if the transfer fails or is canceled.

Example: Receive File * YMODEM Plain "'tempname.com'"

Receive File * YMODEM-G Plain

Purpose: To receive files using the YMODEM-G protocol.

See Receive File * for more information.

Syntax: Receive File * YMODEM-G Plain [ ]

Argument: String - in the expression box, enter the name of the file (optional).

Result: MicroPhone II will receive a single file or a batch of files using the YMODEM-G protocol.

Success flag: Set to True if all files are received successfully.
Set to False if the transfer fails or is canceled.

Example: Receive File * YMODEM-G Plain "'newfile.exe'"

Receive File * ZMODEM Plain

Purpose: To receive files using the ZMODEM protocol.

See Receive File * for more information.

Syntax: Receive File * ZMODEM Plain [ ]

Argument: String - in the expression box, enter the name of the file (optional).

Result: MicroPhone II will receive a single file or a batch of files using the ZMODEM protocol.

Success flag: Set to True if all files are received successfully.
Set to False if the transfer fails or is canceled.

Example: Receive File * ZMODEM Plain "'fallback.bin'"

Remark

Purpose: To include comments in the script listing.

Subcommands: None.

Syntax: Remark [ ]

Argument: String - in the expression box, enter a remark that will annotate your script.

Result: Remarks have no operational significance.

If MicroPhone II is unable to interpret a statement when importing a script, it will read it in as a Remark statement and report the error to you.

Example:

```
1  Remark "This routine checks online mailbox'"
2  Do Script * "'Check Mail'"
```

Remove All Buttons

Purpose: To remove all buttons from the button bar.

Subcommands: None.

Syntax: Remove All Buttons

Argument: None.

Result: MicroPhone II removes all of the buttons from the button bar.

The Install Button ..., Remove Button ... and Remove All Buttons commands allow you to install and replace buttons in mid execution, depending on prompts received from the remote. This simplifies navigation through databases, as well as information search and retrieval.

Example:

```
1  Remove All Buttons
2  Install Button "'Get Mail'"
3  Install Button "'Send Mail'"
4  Install Button "'Log Off'"
```

Remove Button

Purpose: To remove a specific button from the button bar.

Subcommands: None.

Syntax: Remove Button (script name)

Argument: Script - select from the subsidiary list box any script in the settings document.

Result: MicroPhone II removes the button from the bar.

Example:

```
1   If Expression "inbox = empty"
2       Remove Button "'Mail'"
3   End If
```

See also Remove All Buttons.

Repeat

Purpose: Use Repeat with the Until ... command to repeatedly execute a block of statements until a condition is met. The Repeat construct is "static." It should not be used to check for test strings as data is being received, but only when data transmission is halted. To test for new data as it streams into the terminal window, use either the Wait ... or the When ... command.

Repeat is a logical construct. See Logical Constructs for more information.

Subcommands: None.

Syntax: Repeat

Argument: None.

Result: The Repeat and Until ... commands enclose statements that are repeatedly executed as long as the condition defined by the Until ... command is evaluated as False. The first test is made when Until ... is first executed, so a Repeat/Until ... loop will always be executed at least once.

Example: This script will repeatedly try to get the attention of a host (the Telenet network) for up to 3 minutes. To use with TymNet, replace '@' with 'a' and 'INAL=' with the appropriate phrase ('login:').

```
1   Dial Service "'myService'"
2   Set Variable done from Expression "FALSE"
3   Set Variable secs from Expression "0"
4   Repeat
5       When Seconds Have Passed "2"
6           Set Variable secs from Expression "secs+2"
7           Send Text String "'@'"
8           Wait Sixtieths "30"
9           Send Text String "'^M'"
1       Or When Text Equals "'INAL='"
11          Set Variable done from Expression "TRUE"
12      Or When Expression "secs = 120"
13          Hang Up
14          Abort Script
15      End When
16  Until Expression "done"
```

Restore

Purpose: To restore settings to their original values whether they were changed by scripts or from the menu. You may restore all settings, or whatever subset you choose.

Note: Although the settings have been restored, you will still be asked to save settings unless the Restore All Params command is used. These commands do not affect changes to scripts.

Subcommands: Requires one of five subsidiary commands:

Restore All Params
Restore Comm Params
Restore Terminal Params
Restore Text Xfr Params
Restore Protocol Xfr Params

Restore All Params

Purpose: To restore all settings (communications, terminal, text transfer, and protocol transfer) to their original values whether they were changed by scripts or from the menu.

See Restore for more information.

Syntax: Restore All Params

Argument: None.

Result: MicroPhone II reverses all changes made to settings parameters since the last time you saved them. This does not affect changes to scripts.

Example:

```
1  Do Script * "'Log Off'"
2  Restore All Params
```

Restore Comm Params

Purpose: To restore the communications settings to their original values whether they were changed by scripts or from the menu.

See Restore for more information.

Syntax: Restore Comm Params

Argument: None.

Result: MicroPhone II reverses changes made to the communications settings since the last time you saved them.

Restore Terminal Params

Purpose: To restore the terminal settings to their original values whether they were changed by scripts or from the menu.

See Restore for more information.

Syntax: Restore Terminal Params

Argument: None.

Result: MicroPhone II reverses changes made to the terminal settings since the last time the settings document was saved.

Restore Text Xfr Params

Purpose: To restore the text transfer settings to their original values whether they were changed by scripts or from the menu.

See Restore for more information.

Syntax: Restore Text Xfr Params

Argument: None.

Result: MicroPhone II reverses changes made to the text transfer settings since the last time you saved them.

Example:

```
1  Set Text Xfr Param Delay Between Chars "1"
2  Send File * Text Entire File "filename"
3  Restore Text Xfr Params
```

Restore Protocol Xfr Params

Purpose: To restore the protocol transfer settings to their original values whether they were changed by scripts or from the menu.

See Restore for more information.

Syntax: Restore Protocol Xfr Params

Argument: None.

Result: MicroPhone II reverses changes made to the protocol transfer settings since the last time you saved them.

Example:

```
1   If Text Equals "'CONNECT 2400'"
2       Restore Protocol Xfr Params
3   End If
```

Return

Purpose: When a script has been called by another through the Do Script * ... command, a Return ... command will halt the called script and return execution to the calling script.

The calling script resumes execution at the statement following Do Script * ....

If you use the Return ... command in a script that is not called with Do Script * ..., it will act like the Abort Script command.

Subcommands: Requires one of four subsidiary commands:

Return Success
Return Failure
Return Null
Return Expression [ ]

Return Success

Purpose: To halt the current script and return execution to the calling script with the Success flag set to True.

See Return for more information.

Syntax: Return Success

Argument: None.

Result: The called script returns the Success flag as True to the calling script.

Example:

```
1  When Expression "mail2go"
2      Return Success
3  Or When Text Equals "'none'"
4      Chain to Script "'Check Again'"
5  End When
```

Return Failure

Purpose: To halt the current script and return execution to the calling script with the Success flag set to False.

See Return for more information.

Syntax: Return Failure

Argument: None.

Result: The called script returns the Success flag as False to the calling script.

Example:

```
1  When Text Equals "'Address Unknown'"
2      Return Failure
3  Or When Line Contains "'mail sent'"
4      Chain to Script "'Log Off'"
5  End When
```

Return Null

Purpose: To halt the current script and return execution to the calling script without changing the Success flag.

See Return for more information.

Syntax: Return Null

Argument: None.

Result: The called script does not change the Success flag.

Example:

```
1  When Line Contains "'mail'"
2      Return Success
3  Or When Text Equals "'unavailable'"
4      Return Null
5  End When
```

Return Expression

Purpose: To halt the current script and return execution to the calling script with the Success flag set to the expression argument.

See Return for more information.

Syntax: Return Expression [ ]

Argument: Boolean - enter an expression in the expression box that evaluates to True or False.

Result: The called script evaluates the expression, which, if True, sets the Success flag to True, and if False, sets it to False.

Example: Return Expression "Pos (testchar, teststring) > 0"

Script *

Purpose: To import a script from a text file into the current settings document, or export a script into a text file.

Subcommands: Requires one of two subsidiary commands:

Script * Import [ ]
Script * Export [ ]

Script * Import

Purpose: To import a script from a text file into the current settings document.

Syntax: Script * Import [ ]

Argument: Filename - the complete name of the text file containing the script. (Most scripts stored as text files in MicroPhone II include the extension ".SCR".) You must specify a full pathname to the file if it is not in the same directory as the settings document.

Result: The text file is read into MicroPhone II and converted to a script. Any unrecognized lines become remarks. The script will remain in the settings document until the settings document is closed. Note that the script will not be saved unless you choose the Save Settings menu command. Thus, if you close the settings document without saving the script, it will be lost. However, you will be given the opportunity to save it.

Success flag: Set to True if the file is imported (even with errors).
Set to False if the file is not found or if the header is bad and the file was not imported.

Example: 1 Script * Import "'C:\WP\WORD.SCR'"

Note: If a script of the same name already exists in the settings document, the imported script will overwrite it. Statements with syntax errors will be converted to remarks. Even if errors exist, no alerts will be posted because they would interrupt the executing script.

When you write a script that uses both Script * Import and either Chain to Script ... or Do Script * ..., you will have to type the name of the script to be invoked because it won't appear in the subsidiary list box. First, add the Chain to Script or Do Script line without selecting a script name from the subsidiary list box. This line will appear in the listing box with either nothing or the name of one of your existing scripts between the quotation marks. Position your cursor between these marks, delete any existing name and type the script name (not the text filename).

Script * Export

Purpose: To export a script into a text file into the current settings document.

Syntax: Script * Export [ ]

Argument 1: String - in the expression box, enter a string for the name of the script. This must be a script in the current settings document.

Argument 2: String - in the expression box, enter a string for the complete name of the text file in which the script will be saved. (Most scripts stored as text files in MicroPhone II include the extension ".SCR".) You must specify a full pathname to the file if it is not in the same directory as the settings document.

Result: The selected script is converted to text and saved in the text file specified by Argument 2. If the file doesn't exist, it will be created. If the file already exists, it will be overwritten. If no filename is specified in Argument 2, you will be prompted to name the file when the script is executed.

Success flag: Set to True if the script is exported.
Set to False if the file cannot be created, or the script cannot be exported (can't be found, for instance).

Example: Script * Export "'WORD', 'C:\WP\WORD.SCR'"

Select Receive Dir

Purpose: To designate the directory into which files will be received.

Subcommands: None.

Syntax: Select Receive Dir [ ]

Argument: String - in the expression box, enter the name of the folder.

Result: MicroPhone II receives files into the designated directory. If the argument contains an empty string, during script execution MicroPhone II will present you with a dialog box, which allows you to select a directory.

Notes: Incoming files will be received into the designated directory until another directory is chosen (either in a script or from the menu). Your directory choice is saved along with the settings document.

If no directory is designated (either in a script or from the menu), incoming files will be saved into the same directory as the settings document.

Example: Select Receive Dir "'c:\mail'"

Selection *

Purpose: To make a selection in the terminal window from within a script. Once the selection is made, that section of the screen will appear highlighted. The selection may then be saved, printed or used in another command.

Selections in the screen buffer may be made by defining the area with mouse clicks.

Subcommands: Requires one of five subsidiary commands:

Selection Begin
Selection Extend
Selection Save [ ]
Selection Append [ ]
Selection Print

Selection * Begin

Purpose: To mark the beginning of a selection in the terminal window from within a script.

See Selection * for more information

Syntax: Selection * Begin

Argument: None.

Result: MicroPhone II starts selection on the screen at the current position of the cursor.

Success flag: This subcommand does not set the Success flag.

Example:

```
1  Selection * Begin
2  Move Cursor to Absolute Row "1"
3  Move Cursor to Absolute Column "1"
4  Selection * Extend
5  Edit Copy
```

Selection * Extend

Purpose: To mark the end of a selection in the terminal window from within a script.

See Selection * for more information

Syntax: Selection * Extend

Argument: None.

Result: MicroPhone II defines the end of a selection on the screen at the current position of the cursor. The beginning of the selection must have been set by the Selection * Begin command. Selection may be extended below or above the beginning point on the screen.

Success flag: This subcommand does not set the Success flag.

Example:

```
1  Selection * Begin
2  Move Cursor to Absolute Row "24"
3  Move Cursor to Absolute Column "80"
4  Selection * Extend
5  Edit Cut
```

Selection * Save

Purpose: To save a selection from the terminal window to a file.

See Selection * for more information

Syntax: Selection * Save [ ]

Argument: String - in the expression box, enter the name of the file to which you want to save the selection.

Result: MicroPhone II saves selected data to the file. On execution, if the named file already exists, its contents will be overwritten by the selection. If no file is named, MicroPhone II prompts you to select one.

Success flag: Set to True if the selection is successfully written to the file.
Set to False if the selection cannot be written to the file.

Example: Selection * Save "'Data' & hour & minute"

Selection * Append

Purpose: To add a selection in the terminal window to a file.

See Selection * for more information

Syntax: Selection * Append [ ]

Argument: String - in the expression box, enter the name of the file to which you want to append the selection.

Result: MicroPhone II adds selected data to an already existing file. If the named file does not exist, the program creates it. If no file is named, MicroPhone II prompts you to select one.

Success flag: Set to True if the selection is successfully written to the file.
Set to False if the selection cannot be written to the file.

Example: Selection * Append '"LogFile"'

Selection * Print

Purpose: To print a selection in the terminal window.

See Selection * for more information

Syntax 5: Selection * Print

Argument: None.

Result: MicroPhone II prints the selected data. The printer must be connected and on-line. Current printer setup will govern the print job.

Success flag: This subcommand does not set the Success flag.

Example:

```
1  When Button "'Screen Dump'"
2       Move Cursor to Absolute Row "1"
3       Move Cursor to Absolute Column "1"
4       Selection * Begin
5       Move Cursor to Absolute Row "24"
6       Move Cursor to Absolute Column "80"
7       Selection * Extend
8       Selection * Print
9       Do Script * "'Restore Cursor'"
10 Or When Seconds Have Passed "2"
11 End When
```

Send File *

Purpose: To send files using any of the file transfer protocols supported by MicroPhone II.

Transfers are governed by your current Text Transfer Settings and Protocol Transfer Settings.

If you do not include a file name in the Send File * ... statement, MicroPhone II prompts you for a name during script execution. Canceling the prompt will cancel the transfer, and the Success flag will be set to False.

If you do not provide a full pathname in the expression box, MicroPhone II will look for the file in the directory specified by the command Select Receive Directory in the Transfer menu or by the Select Receive Dir ... script command.

Subcommands: Requires one of ten subsidiary commands:

Send File * Text Entire File [ ]
Send File * Text Line by Line [ ]
Send File * Text Close
Send File * XMODEM Plain [ ]
Send File * YMODEM Plain [ ]
Send File * YMODEM Batch Plain [ ]
Send File * ZMODEM Binary [ ]
Send File * ZMODEM ASCII [ ]
Send File * ZMODEM Batch Binary [ ]
Send File * ZMODEM Batch ASCII [ ]

Send File * Text Entire File

Purpose: To send files using text transfer mode.

See Send File * for more information.

Syntax: Send File * Text Entire File [ ]

Argument: String - in the expression box, enter the name of the file to send.

Result: Text is sent as if it had been typed from the keyboard, without any error-checking protocol.

Success flag: Set to True if the file is sent successfully.
Set to False if the transfer fails or is canceled.

Send File * Text Line by Line

Purpose: To open a file from which text lines can be sent using the Send Line * command.

See Send File * for more information.

Syntax: Send File * Text Line by Line [ ]

Argument: String - in the expression box, enter the name of the text file.

Result: The Send File * Text Line by Line command will not cause any data to be sent. Its function is solely to open a file from which selected lines can be sent - or assigned to variables - using the Send Line * and Skip Line * commands.

A new line will be sent (from the first line of the file to the last, one line at a time) each time the Send Line * command is encountered.

Note: Use the File * ... commands and related functions wherever possible. The Send File * Text Line by Line, Send Line * and Skip Line * commands are included only to support past versions of MicroPhone II.

Success flag: Set to True if the file is successfully opened.
Set to False if the file cannot be opened.

Send File * Text Close

Purpose: To close the file opened using the Send File Text Line by Line * command.

See Send File * for more information.

Syntax: Send File * Text Close

Argument: None.

Result: MicroPhone II closes the file opened using the Send File Text Line by Line * script command.

Success flag: Set to True if the file is sent successfully.
Set to False if the transfer fails or is canceled.

Send File * XMODEM Plain

Purpose: To send a file using the XMODEM protocol.

See Send File * for more information.

Syntax: Send File * XMODEM Plain [ ]

Argument: String - in the expression box, enter the name of the file to send.

Result: MicroPhone II sends the file using the XMODEM protocol.

Success flag: Set to True if the file is sent successfully.
Set to False if the transfer fails or is canceled.

Example: Send File * XMODEM Plain "'Graphics.TIF'"

Send File * YMODEM Plain

Purpose: To send a file using the YMODEM protocol.

See Send File * for more information.

Syntax: Send File * YMODEM Plain [ ]

Argument: String - in the expression box, enter the name of the file to send.

Result: MicroPhone II sends the file using the YMODEM protocol. (Because YMODEM-G is the choice of the receiver, this same command should also be used to send a file in YMODEM-G protocol.)

Success flag: Set to True if the file is sent successfully.
Set to False if the transfer fails or is canceled.

Example: Send File * YMODEM Plain '"C:\Graphic\Paint.fil"'

Send File * YMODEM Batch Plain

Purpose: To send a batch of files using the YMODEM protocol.

See Send File * for more information.

Syntax: Send File * YMODEM Batch Plain [ ]

Argument: String - in the expression box, enter the name of the batch to send.

Result: MicroPhone II sends the batch using the YMODEM protocol. (Because YMODEM-G is the choice of the receiver, this same command should also be used to send a batch in YMODEM-G protocol.)

Success flag: Set to True if at least one of the files in the batch is sent successfully and the transfer does not fail and is not canceled.

Set to False if none of the files in the batch could be found or opened, or if the transfer fails or is canceled.

Example: Send File * YMODEM Batch Plain "'Samples.MBT'"

Send File * ZMODEM Binary

Purpose: To send a file using the ZMODEM protocol.

See Send File * for more information.

Syntax: Send File * ZMODEM Binary [ ]

Argument: String - in the expression box, enter the name of the file to send.

Result: MicroPhone II sends the file using the ZMODEM protocol.

Success flag: Set to True if the file is sent successfully.
Set to False if the transfer fails or is canceled.

Send File * ZMODEM ASCII

Purpose: To send a file using the ZMODEM ASCII protocol.

See Send File * for more information.

Syntax: Send File * ZMODEM ASCII [ ]

Argument: String - in the expression box, enter the name of the file to send.

Result: MicroPhone II sends the file using the ZMODEM ASCII protocol. End-of-line characters are converted to match those required by the receiving computer.

Success flag: Set to True if the file is sent successfully.
Set to False if the transfer fails or is canceled.

Send File * ZMODEM Batch Binary

Purpose: To send a batch of files using the ZMODEM protocol.

See Send File * for more information.

Syntax: Send File * ZMODEM Batch Binary [ ]

Argument: String - in the expression box, enter the name of the batch to send.

Result: MicroPhone II sends the batch using the ZMODEM protocol.

Success flag: Set to True if at least one of the files in the batch is sent successfully and the transfer does not fail and is not canceled.

Set to False if none of the files in the batch could be found or opened, or if the transfer fails or is canceled.

Send File * ZMODEM Batch ASCII

Purpose: To send a batch of files using the ZMODEM ASCII protocol.

See Send File * for more information.

Syntax: Send File * ZMODEM Batch ASCII [ ]

Argument: String - in the expression box, enter the name of the batch to send.

Result: MicroPhone II sends the batch using the ZMODEM ASCII protocol. End-of-line characters are converted to match those required by the receiving computer.

Success flag: Set to True if at least one of the files in the batch is sent successfully and the transfer does not fail and is not canceled.

Set to False if none of the files in the batch could be found or opened, or if the transfer fails or is canceled.

Send Line *

Purpose: To send the next line of the text file opened by the Send File Text Line by Line * command.

Note: Use the File * ... commands and related functions wherever possible. The Send File Text Line by Line *, Send Line * and Skip Line * commands are included only to support past versions of MicroPhone II.

Subcommands: None.

Syntax: Send Line *

Argument: None.

Result: Each time this command is executed, it will send the next line of the text file opened by the Send File Text Line by Line * command.

A combination of the three commands - Send File Text Line by Line *, Send Line * and Skip Line * - allows you to send selected lines from a text file.

Success flag: Set to True each time MicroPhone II sends a line.
Set to False when the end of the file is reached.

Example:

```
1  Send File * Text Line by Line "'Sample'"
2  Repeat
3      Send Line *
4  Until Failure
```

Send Local

Purpose: To send characters to your screen, the capture file, or the printer, without sending them to the remote computer.

Note: You may use Send Local ... to customize your screen display. In particular, you can use the function NextCh, to watch for incoming characters before they are displayed on the screen, convert them at will, and then draw the screen to your specifications with the use of screen addressing and Send Local to Screen script commands.

Subcommands: Requires one of four subsidiary commands:

Send Local to Screen [ ]
Send Local to File [ ]
Send Local to Screen & File [ ]
Send Local to Printer [ ]

Send Local to Screen

Purpose: To send characters to your screen without sending them to the remote computer.

See Send Local for more information.

Syntax: Send Local to Screen [ ]

Argument: String - in the expression box, enter the text to send to the screen.

Result: MicroPhone II displays on the screen the data entered in the expression box, starting at the current cursor position.

Example:
```
1  Set Variable * nxtchar from Expression "NextKey"
2  Send Local to Screen "nxtchar"
```

Send Local to File

Purpose: To send characters to the capture file without sending them to the remote computer.

See Send Local for more information.

Syntax: Send Local to File [ ]

Argument: String - in the expression box, enter the text to send to the capture file.

Result: MicroPhone II sends the data entered in the expression box to the capture file that is currently open.

Example:
```
1  Set Variable * stamp from Date & Time
2  Send Local to File "'Logged in at ' & stamp"
```

Send Local to Screen & File

Purpose: To send characters to both your screen and the capture file without sending them to the remote computer.

See Send Local for more information.

Syntax: Send Local to Screen & File [ ]

Argument: String - in the expression box, enter the text to send to the screen and to the capture file.

Result: MicroPhone II displays the data entered in the expression box on the screen starting at the current cursor position, and then sends the text to the capture file that is currently open.

Example: Send Local to Screen & File "'File received at ' & stamp & 'from ' & servicename"

Send Local to Printer

Purpose: To send characters to the printer without sending them to the remote computer.

See Send Local for more information.

Syntax: Send Local to Printer [ ]

Argument: String - in the expression box, enter the text to send to the printer.

Result: MicroPhone II sends the data you enter in the expression box to the printer.

Example:
```
1  Set Variable * formfeed from Expression "'^L'"
2  Send Local to Printer "formfeed"
```

Send Text

Purpose: To send text strings to the remote computer.

Subcommands: Requires one of six subsidiary commands:

Send Text String [ ]
Send Text String (protected) [ ]
Send Text from File [ ]
Send Text from File (protected) [ ]
Send Text from Dialog Box [ ]
Send Text from Dialog Box (prot.) [ ]

Send Text String

Purpose: To send text strings to the remote computer.

Syntax: Send Text String [ ]

Argument: String - in the expression box, enter the text to send.

Result: MicroPhone II sends the string as if it were typed from the keyboard. The delay between characters, set in the Text Transfer Settings, applies to this command.

Example: Send Text String "'logout^M'"

Send Text String (protected)

Purpose: To send text strings to the remote computer while hiding them at the local computer.

Syntax: Send Text String (protected) [ ]

Argument: String - in the expression box, enter the text to send.

Remember to enclose protected text strings with single quotes.

Result: MicroPhone II sends the string as if it were typed from the keyboard. The characters will be visually replaced by the string "<*<*<Protectednn>*>*>" (where nn represents the number of protected strings in your script - up to 20) whenever the script is reviewed.

To protect the confidentiality of the data being sent, the program automatically turns Local Echo off, sends the protected data and then turns it back on again.

This command should be used when you wish to maintain the secrecy of such things as passwords from anyone who may have access to the disk. However, you should remember that, once hidden, passwords can't be retrieved.

Example: Send Text String (protected) "<*<*<Protected01>*>*>"

Send Text from File

Purpose: To send a text line from a file to the remote computer.

Syntax: Send Text from File [ ]

Argument: String - in the expression box, enter the name of a file.

Result: MicroPhone II sends the first line, up to 255 characters, from the named text file as if text were typed from the keyboard.

If, on script execution, no such file as named in the expression box exists, the script will prompt you for a text string to send using the file name as a prompt in a dialog box. The script will then send whatever you type, up to 255 characters, and save it to disk as a file with the name you have designated.

Example: Send Text from File "'Company.fil'"

Send Text from File (protected)

Purpose: To send a text line from a file to the remote computer while hiding at the local computer.

Syntax: Send Text from File (protected) [ ]

Argument: String - in the expression box, enter the name of a file.

Result: MicroPhone II sends the first line, up to 255 characters, from the named text file as if text were typed from the keyboard.

If no such file exists, the script will prompt you for a text string to send using the filename as a prompt in a dialog box. The script will then send whatever you type, up to 255 characters, and save it to disk as a hidden file with the designated name. The file will not appear in any directory listing and you may not delete it (except in Windows File Manager).

To protect the confidentiality of the data being sent, the program automatically turns Local Echo off, sends the data and then turns it back on again.

Example: Send Text from File (protected) '"Passkey"'

Send Text from Dialog Box

Purpose: To send a text line from a dialog box entry to the remote computer.

Syntax: Send Text from Dialog Box [ ]

Argument: String - in the expression box, enter a prompt for the dialog box.

Result: MicroPhone II prompts you with a dialog box, where you can type text, up to 255 characters long. When you click the dialog box's OK button, text will be sent. If you click OK with no data entered, no text will be sent.

Example: Send Text from Dialog Box "'Please enter your ID: '"

Send Text from Dialog Box (prot.)

Purpose: To send a text line from a dialog box entry to the remote computer while hiding at the local computer.

Syntax: Send Text from Dialog Box (prot.) [ ]

Argument: String - in the expression box, enter a prompt for the dialog box.

Result: MicroPhone II prompts you with the dialog box, where you can type text, up to 255 characters. As you type, you will see only block characters.

When you click the dialog box's OK button, the actual text will be sent. If you click OK with no data entered, no text will be sent. To protect the confidentiality of the data being sent, the program will automatically turn Local Echo off, send the data and then turn it back on again.

Example: Send Text from Dialog Box (prot.) "'Please enter your password: '"

Set Comm Param

Purpose: To change the communications parameters.

Subcommands: Requires one of nine subsidiary commands:

Set Comm Param Baud Rate
Set Comm Param Bits per Character
Set Comm Param Parity
Set Comm Param Stop Bits
Set Comm Param Connection Port
Set Comm Param Modem Driver [ ]
Set Comm Param Connection Driver [ ]
Set Comm Param Flow Control
Set Comm Param Character Set [ ]

Set Comm Param Baud Rate

Purpose: To change the communications parameter baud rate.

Syntax: Set Comm Param Baud Rate

Options: 50, 75, 110, 134.5, 150, 200, 300, 450, 600, 1,200, 1,800, 2,000, 2,400, 3,600, 4,800, 7,200, 9,600, 19,200, 38,400, 57,600 and 115,200

Argument: None.

Set Comm Param Bits per Character

Purpose: To change the communications parameter bits per character.

Syntax: Set Comm Param Bits per Character

Options: 5, 6, 7 and 8 - select one from the third list box.

Argument: None.

Set Comm Param Parity

Purpose: To change the communications parameter parity.

Syntax: Set Comm Param Parity

Options: None, Odd, Even, Mark and Space - select one from the third list box.

Argument: None.

Set Comm Param Stop Bits

Purpose: To change the communications parameter stop bits.

Syntax: Set Comm Param Stop Bits

Options: Auto, 1, 1.5 and 2 - select one from the third list box.

Argument: None.

Set Comm Param Connection Port

Purpose: To change the communications parameter connection port.

Syntax: Set Comm Param Connection Port

Options: Com1 - Com4, None, and others - select one from the third list box.

Note: This command overrides the setting of the physical connection port in the Connection Driver. (The connection port is normally set using the Save Connection Driver as command in the Script Manager's File menu.) It should be used only within a connection driver. The connection driver scripts must be written to handle the differences between the physical connection ports set using this command.

To change access to a different logical connection port, use Set Comm Param Connection Driver.

Argument: None.

Set Comm Param Modem Driver

Purpose: To change the communications parameter modem driver.

Syntax: Set Comm Param Modem Driver [ ]

Argument: String - in the expression box, enter the name of an available modem driver.

Set Comm Param Connection Driver

Purpose: To change the communications parameter connection driver.

Syntax: Set Comm Param Connection Driver [ ]

Argument: String - in the expression box, enter the name of an available connection driver.

Set Comm Param Flow Control

Purpose: To change the communications parameter flow control.

Syntax: Set Comm Param Flow Control

Options: X-on/X-off, Hardware and None - select one from the third list box.

Argument: None.

Set Comm Param Character Set

Purpose: To change the communications parameter character set.

Syntax: Set Comm Param Character Set [ ]

Argument: String - in the expression box, enter the name of an available character set.

Set Prtcl Xfr Param

Purpose: To change the protocol file transfer parameters.

Subcommands: Requires one of ten subsidiary commands:

Set Prtcl Xfr Param CRC
Set Prtcl Xfr Param 1K Blocks
Set Prtcl Xfr Param XMODEM Mode
Set Prtcl Xfr Param Auto-detect
Set Prtcl Xfr Param Timeout [ ]
Set Prtcl Xfr Param ZMODEM Packet Size
Set Prtcl Xfr Param ZMODEM Window Size [ ]
Set Prtcl Xfr Param ZMODEM CRC Size
Set Prtcl Xfr Param ZMODEM Resurrect
Set Prtcl Xfr Param ZMODEM Esc Ctrl Chars

Set Prtcl Xfr Param CRC

Purpose: To change the protocol file transfer parameter CRC.

Syntax: Set Prtcl Xfr Param CRC

Options: No and Yes - select one from the third list box.

Argument: None.

Note: This command applies only to the XMODEM protocol.

Set Prtcl Xfr Param 1K Blocks

Purpose: To change the protocol file transfer parameter block size.

Syntax: Set Prtcl Xfr Param 1K Blocks

Options: Automatic, On and Off - select one from the third list box.

Argument: None.

Set Prtcl Xfr Param XMODEM Mode

Purpose: To change the protocol file transfer parameter mode.

Syntax: Set Prtcl Xfr Param XMODEM Mode

Options: Text, XMODEM, YMODEM, YMODEM-G and ZMODEM - select one from the third list box.

Argument: None.

Set Prtcl Xfr Param Auto-detect

Purpose: To change the protocol file transfer parameter Auto-detect.

Syntax: Set Prtcl Xfr Param Auto-detect

Options: No and Yes - select one from the third list box.

Argument: None.

Note: This command applies only to the ZMODEM protocol.

Set Prtcl Xfr Param Timeout

Purpose: To change the protocol file transfer parameter timeout.

Syntax: Set Prtcl Xfr Param Timeout [ ]

Argument: Integer - in the expression box, enter the number of seconds (no less than 10).

Note: This command applies to all protocols.

Set Prtcl Xfr Param ZMODEM Packet Size

Purpose: To change the protocol file transfer parameter ZMODEM packet size.

Syntax: Set Prtcl Xfr Param ZMODEM Packet Size

Options: 128, 512 and 1024 - select one from the third list box.

Argument: None.

Set Prtcl Xfr Param ZMODEM Window Size

Purpose: To change the protocol file transfer parameter ZMODEM window size.

Syntax: Set Prtcl Xfr Param ZMODEM Window Size [ ]

Argument: Integer - in the expression box, enter the number of bytes. Zero indicates ZMODEM will not wait for a NAK (unlimited window size).

Note: This setting should be zero or a multiple of the packet size.

Set Prtcl Xfr Param ZMODEM CRC Size

Purpose: To change the protocol file transfer parameter ZMODEM CRC size.

Syntax: Set Prtcl Xfr Param ZMODEM CRC Size

Options: CRC-16 and CRC-32 - select one from the third list box.

Argument: None.

Set Prtcl Xfr Param ZMODEM Resurrect

Purpose: To change the protocol file transfer parameter ZMODEM resurrect.

Syntax: Set Prtcl Xfr Param ZMODEM Resurrect

Options: No and Yes - select one from the third list box.

Argument: None.

Set Prtcl Xfr Param ZMODEM Esc Ctrl Chars

Purpose: To change the protocol file transfer parameter ZMODEM escape control characters.

Syntax: Set Prtcl Xfr Param ZMODEM Esc Ctrl Chars

Options: No and Yes - select one from the third list box.

Argument: None.

Set Term Param

Purpose: To change the terminal parameters.

Subcommands: Requires one of thirteen subsidiary commands:

Set Term Param Type
Set Term Param Cursor Shape
Set Term Param Backspace Key
Set Term Param Local Echo
Set Term Param Auto Wraparound
Set Term Param New Line
Set Term Param Columns [ ]
Set Term Param Rows [ ]
Set Term Param Font Size [ ]
Set Term Param Answerback Message [ ]
Set Term Param Capture on CR
Set Term Param Capture on Clear
Set Term Param Strip 8th Bit

Set Term Param Type

Purpose: To change the terminal parameter terminal type.

Syntax: Set Term Param Type

Options: Select a terminal type from the third list box.

Argument: None.

Set Term Param Cursor Shape

Purpose: To change the terminal parameter cursor shape.

Syntax: Set Term Param Cursor Shape

Options: Block and Underline - select one from the third list box.

Argument: None.

Set Term Param Backspace Key

Purpose: To change the terminal parameter backspace key.

Syntax: Set Term Param Backspace Key

Options: Backspace, Delete and Correction - select one from the third list box.

Argument: None.

Set Term Param Local Echo

Purpose: To change the terminal parameter local echo.

Syntax: Set Term Param Local Echo

Options: No and Yes - select one from the third list box.

Argument: None.

Set Term Param Auto Wraparound

Purpose: To change the terminal parameter auto wraparound.

Syntax: Set Term Param Auto Wraparound

Options: No and Yes - select one from the third list box.

Argument: None.

Set Term Param New Line

Purpose: To change the terminal parameter new line.

Syntax: Set Term Param New Line

Options: No and Yes - select one from the third list box.

Argument: None.

Set Term Param Columns

Purpose: To change the terminal parameter columns.

Syntax: Set Term Param Columns [ ]

Argument: Enter the column width in the expression box.

Set Term Param Rows

Purpose: To change the terminal parameter rows.

Syntax: Set Term Param Rows [ ]

Argument: Enter the row height in the expression box.

Set Term Param Font Size

Purpose: To change the terminal parameter font size.

Syntax: Set Term Param Font Size [ ]

Argument: Enter the font size in the expression box. This must be a size supported by the currently chosen font.

Set Term Param Answerback Message

Purpose: To change the terminal parameter answerback message.

Syntax 10: Set Term Param Answerback Message [ ]

Argument: String - the response to an Enquire character. The Answerback Message may be as long as 255 characters.

Set Term Param Capture on CR

Purpose: To change the terminal parameter capture on carriage return.

Syntax: Set Term Param Capture on CR

Options: No and Yes - select one from the third list box.

Argument: None.

Set Term Param Capture on Clear

Purpose: To change the terminal parameter capture on clear.

Syntax: Set Term Param Capture on Clear

Options: No and Yes - select one from the third list box.

Argument: None.

Set Term Param Strip 8th Bit

Purpose: To change the terminal parameter strip 8th bit.

Syntax: Set Term Param Strip 8th Bit

Options: No and Yes - select one from the third list box.

Argument: None.

Set Text Xfr Param

Purpose: To change the text transfer parameters.

Subcommands: Requires one of nine subsidiary commands:

Set Text Xfr Param X-on/X-off Sending
Set Text Xfr Param X-on/X-off Receiving
Set Text Xfr Param Wait for Echo
Set Text Xfr Param Wait for Prompt Char [ ]
Set Text Xfr Param Delay between Chars [ ]
Set Text Xfr Param Delay between Lines [ ]
Set Text Xfr Param Word-wrap at Column [ ]
Set Text Xfr Param End Outgoing Lines with
Set Text Xfr Param End Captured Lines with

Set Text Xfr Param X-on/X-off Sending

Purpose: To change the text transfer parameter X-on/X-off while sending.

Syntax: Set Text Xfr Param X-on/X-off Sending

Options: No and Yes - select one from the third list box.

Argument: None.

Set Text Xfr Param X-on/X-off Receiving

Purpose: To change the text transfer parameter X-on/X-off while receiving.

Syntax: Set Text Xfr Param X-on/X-off Receiving

Options: No and Yes - select one from the third list box.

Argument: None.

Set Text Xfr Param Wait for Echo

Purpose: To change the text transfer parameter wait for echo.

Syntax: Set Text Xfr Param Wait for Echo

Options: None, CR, LF and All - select one from the third list box.

Argument: None.

Set Text Xfr Param Wait for Prompt Char

Purpose: To change the text transfer parameter wait for prompt character.

Syntax: Set Text Xfr Param Wait for Prompt Char [ ]

Argument: String - in the expression box, enter one character.

Set Text Xfr Param Delay between Chars

Purpose: To change the text transfer parameter delay between characters.

Syntax: Set Text Xfr Param Delay between Chars [ ]

Argument: Integer - in the expression box, enter the time in a 1/60th-of-a-second increment.

Set Text Xfr Param Delay between Lines

Purpose: To change the text transfer parameter delay between lines.

Syntax: Set Text Xfr Param Delay between Lines [ ]

Argument: Integer - in the expression box, enter the time in a 1/60th-of-a-second increment.

Set Text Xfr Param Word-wrap at Column

Purpose: To change the text transfer parameter word-wrap at column.

Syntax: Set Text Xfr Param Word-wrap at Column [ ]

Argument: Integer - in the expression box, enter the number of characters per line.

Set Text Xfr Param End Outgoing Lines with

Purpose: To change the text transfer parameter outgoing lines ending character sequence

Syntax: Set Text Xfr Param End Outgoing Lines with

Options: Nothing, CR, LF, CR & LF and LF & CR - select one from the third list box.

Argument: None.

Set Text Xfr Param End Captured Lines with

Purpose: To change the text transfer parameter captured lines ending character sequence.

Syntax: Set Text Xfr Param End Captured Lines with

Options: Nothing, CR, LF, CR & LF and LF & CR - select one from the third list box.

Set Variable *

Purpose: To create and assign values to variables.

The program automatically creates a variable when it first assigns a value to it, so you don't need to declare it separately. Variables retain their values until you delete them or quit MicroPhone II.

All variables are global to scripts, including modem scripts.

A variable name may contain up to 24 characters; it must begin with an alphabetical character but may contain digits after the initial character. The only characters that may be used are the letters A through Z, the underscore and the digits 1 through 9. Variable names are not case sensitive; therefore, VAR1 and var1 are considered to be the same variable. A variable may be given any name except for those names already taken by functions, operators, or names that make up a script command (e.g., file, line, text, etc.).

The Set Variable * ... command displays a box to the right of the script command list box labeled Variable: where the variable name is entered.

Subcommands: Requires one of eight subsidiary commands:

Set Variable * (variable name) from Expression [ ]
Set Variable * (variable name) from Dialog Box [ ]
Set Variable * (variable name) from Dialog Box (prot.) [ ]
Set Variable * (variable name) from File Line
Set Variable * (variable name) from File Dialog [ ]
Set Variable * (variable name) from New File Dialog [ ]
Set Variable * (variable name) from Date & Time
Set Variable * (variable name) from Selection

Set Variable * from Expression

Purpose: To assign the value of an expression to a variable.

See Set Variable * for more information.

Syntax: Set Variable * (variable name) from Expression [ ]

Argument: Expression - in the expression box, enter any expression.

See Expressions for rules on expressions.

Result: MicroPhone II evaluates the expression and assigns its value to the variable.

Success flag: Set to True if the expression was evaluated and the value was assigned to the variable.

Set to False if the expression is not evaluated or the value was not assigned to the variable. Failure can occur because a function in the expression returns False, the variable name was not valid or no memory was available for the variable.

Example: Set Variable * teststring from Expression "TheLine (CursorRow - 1)"

Set Variable * from Dialog Box

Purpose: To assign the entry in a dialog box to a variable.

See Set Variable * for more information.

Syntax: Set Variable * (variable name) from Dialog Box [ ]

Argument: String - in the expression box, enter the prompt for the dialog box.

Result: During script execution, MicroPhone II will present a dialog box where you should enter the value to assign to the variable. If you enter MicroPhone II's standard notation for a control character, the symbols will be interpreted as a control character. For example, "^I" is interpreted as a tab.

Success flag: Set to True if the expression was evaluated and the value was assigned to the variable.

Set to False if the expression is not evaluated or the value was not assigned to the variable. Failure can occur because a function in the expression returns False, the variable name was not valid or no memory was available for the variable.

Example: Set Variable * id from Dialog Box "'What is your ID?'"

Set Variable * from Dialog Box (prot.)

Purpose: To assign the entry in a dialog box to a variable, while hiding the entry on the local computer.

See Set Variable * for more information.

Syntax: Set Variable * (variable name) from Dialog Box (prot.) [ ]

Argument: String - in the expression box, enter the prompt for the dialog box.

Result: During script execution, MicroPhone II will present a dialog box where you should enter the value to assign to the variable. If you enter MicroPhone II's standard notation for a control character, the symbols will be interpreted as a control character. For example, "^I" is interpreted as a tab.

Data typed in this dialog box is displayed as blocks. When you click the dialog box's OK button, that data will be assigned to the named variable.

Success flag: Set to True if the expression was evaluated and the value was assigned to the variable.

Set to False if the expression is not evaluated or the value was not assigned to the variable. Failure can occur because a function in the expression returns False, the variable name was not valid or no memory was available for the variable.

Example: Set Variable * password from Dialog Box (prot.) "'What is your password?'"

Set Variable * from File Line

Purpose: To assign one line of text from a file to a variable.

See Set Variable * for more information.

Syntax: Set Variable * (variable name) from File Line

Argument: None.

Result: MicroPhone II assigns a line of a text file, specified by a Send File * Text Line by Line command, to the named variable. Use the Skip Line * command to skip over lines to reach the desired line. The assigned line may consist of zero to 255 characters and ends with either the 255th character or a carriage return, whichever comes first.

Success flag: Set to True if the expression was evaluated and the value was assigned to the variable.

Set to False if the expression is not evaluated or the value was not assigned to the variable. Failure can occur because a function in the expression returns False, the variable name was not valid or no memory was available for the variable.

Note that a variable may be set to null at the same time the Success flag is set to True (as when an empty line is encountered in the middle of the file).

Example:

```
1  Set Variable * search from File Line
2  Do Script * "'Dow Jones'"
```

Set Variable * from File Dialog

Purpose: To assign the name of a file to a variable.

See Set Variable * for more information.

Syntax: Set Variable * (variable name) from File Dialog [ ]

Argument: String - in the expression box, enter the prompt for the dialog box.

Result: During script execution, MicroPhone II lets you select any file on any disk. It then assigns to the variable the full name of the file (including the disk drive name and any path information). Note that the program does not open the file or affect it in any way.

If you click the Cancel button in the dialog box, the variable remains unchanged.

Success flag: Set to True if the expression was evaluated and the value was assigned to the variable.

Set to False if the expression is not evaluated or the value was not assigned to the variable. Failure can occur because a function in the expression returns False, the variable name was not valid or no memory was available for the variable.

Example: To append to a capture file that you will select when the script is running, include the script commands:

```
1  Set Variable * capname from File Dialog
2  Append to File * "capname"
```

Set Variable * from New File Dialog

Purpose: To assign a new file name to a variable.

See Set Variable * for more information.

Syntax: Set Variable * (variable name) from New File Dialog [ ]

Argument: String - in the expression box, enter the prompt for the dialog box.

Result: During script execution, MicroPhone II displays a dialog box that lets you enter a new name for a file. The program then assigns to the variable the full name of the file (including the disk name and any path information). Note that MicroPhone II does not create the new file.

If you click the Cancel button in the dialog box, the variable remains unchanged.

Success flag: Set to True if the expression was evaluated and the value was assigned to the variable.

Set to False if the expression is not evaluated or the value was not assigned to the variable. Failure can occur because a function in the expression returns False, the variable name was not valid or no memory was available for the variable.

Example:

```
1  Set Variable * capname from New File Dialog
2  Open Capt. File * "capname"
```

Set Variable * from Date & Time

Purpose: To assign the current date and time to a variable.

See Set Variable * for more information.

Syntax: Set Variable * (variable name) from Date & Time

Argument: None.

Result: This command assigns to the variable the current date and time. The format is: mm/dd/yy hh:mm:ss. For example, it might be set to 2/28/88 14:23:37.

Success flag: Set to True if the expression was evaluated and the value was assigned to the variable.

Set to False if the expression is not evaluated or the value was not assigned to the variable. Failure can occur because a function in the expression returns False, the variable name was not valid or no memory was available for the variable.

Example: To enter the time and date in capture files, use the following statements:

```
1  Append to File * "'CompuSrv.txt'"
2  Set Variable* DT from Date & Time
3  Send Local to File   "'----------'"
4  Send Local to File   "'Added at '&DT"
```

Set Variable * from Selection

Purpose: To assign highlighted data to a variable.

See Set Variable * for more information.

Syntax: Set Variable * (variable name) from Selection

Argument: None.

Result: This command assigns to the variable the data that has been selected in the terminal window. You may make a selection with the mouse or from a script with the appropriate screen addressing and editing commands.

Success flag: Set to True if the expression was evaluated and the value was assigned to the variable.

Set to False if the expression is not evaluated or the value was not assigned to the variable. Failure can occur because a function in the expression returns False, the variable name was not valid or no memory was available for the variable.

Example:

```
1  Selection * Begin
2  Move Cursor to Absolute Column "30"
3  Selection * Extend
4  Set Variable * addressee from Selection
```

Settings

Purpose: To save or close an existing settings document, or to open a new settings document.

Note: When used in conjunction with DDE, this command will allow another application to control multiple MicroPhone II sessions.

Subcommands: Requires one of three subsidiary commands:

Settings Open [ ]
Settings Close
Settings Save [ ]

Settings Open

Purpose: To open a new settings document.

Note: When used in conjunction with DDE, this command will allow another application to control multiple MicroPhone II sessions.

Syntax: Settings Open [ ]

Argument 1: String - In the expression box, enter the name of the new settings document to be opened.

Argument 2: Boolean (optional) - In the expression box, enter TRUE if the current settings document should be closed.

Result: The settings document specified by Argument 1 is opened. If argument 2 is TRUE, the current settings document is also closed. If Argument 2 is not provided, the current settings document will remain open.

If the document specified by Argument 1 is not found among MicroPhone II's list of installed document names, MicroPhone II attempts to open a settings document using Argument 1 as a filename. If that fails, MicroPhone II continues execution with the next script command.

Settings Close

Purpose: To close a settings document.

Note: When used in conjunction with DDE, this command will allow another application to control multiple MicroPhone II sessions.

Syntax: Settings Close

Argument: None.

Result: The current settings document is closed.

If this command is executed from a script, no commands after it in the script will be executed.

Settings Save

Purpose: To save a settings document.

Note: When used in conjunction with DDE, this command will allow another application to control multiple MicroPhone II sessions.

Syntax: Settings Save [ ]

Argument 1: String - In the expression box, enter the document name used to store the settings document.

Argument 2: String - In the expression box, enter the filename used to store the settings document.

Result: The current settings document is saved using the document name specified in Argument 1 into the file specified in Argument 2.

If Argument 1 is not provided, the current document name is used. If Argument 2 is not provided, a unique filename will be created. (See the discussion of Save Settings as under the File Menu.) If the specified file already exists, it will be overwritten.

Signal

Purpose: To send a control signal to the attached connection hardware, such as a modem or a network modem server's command interpreter.

Subcommands: Requires one of seven subsidiary commands:

Signal Short Break
Signal Long Break
Signal Assert DTR
Signal Negate DTR
Signal Flash DTR
Signal Hold
Signal Send IOCTL [ ]

Signal Short Break

Purpose: To send a short break signal to the attached connection hardware.

Syntax: Signal Short Break

Argument: None.

Result: MicroPhone II sends a short (233 milliseconds) Break signal.

Signal Long Break

Purpose: To send a long break signal to the attached connection hardware.

Syntax: Signal Long Break

Argument: None.

Result: MicroPhone II sends a long (3.5 seconds) Break signal.

Note: Use the following key sequences to issue a break from the terminal window:

Short Break (233 milliseconds) Ctrl-break or Alt `

Long Break (3.5 seconds) Ctrl-Shift break or Alt ~

Signal Assert DTR

Purpose: To set DTR on the attached modem.

Syntax: Signal Assert DTR

Argument: None.

Result: The voltage on the modem's DTR pin is set high.

Signal Negate DTR

Purpose: To turn off DTR on the attached modem.

Syntax: Signal Negate DTR

Argument: None.

Result: The voltage on the modem's DTR pin is set low.

Signal Flash DTR

Purpose: To flash DTR on the attached modem.

Syntax: Signal Flash DTR

Argument: None.

Result: The voltage on the DTR pin on the modem is set low for one second, and then set high.

Signal Hold

Purpose: To suspend the current NASI connection.

Syntax: Signal Hold

Argument: None.

Result: The connection to a network modem is placed on hold. You may then type commands directly to the NASI Command Interpreter. See your NACS documentation for NASI Command Interpreter specific commands.

This command will not do anything unless a connection to a Novell NetWare Asynchronous Communications Server port is open.

Example: To place the current connection on hold, list the status of all connections and disconnect the first connection on hold; then use the following script:

```
1   Signal Hold
2   Set Variable * circuit from expression "-1"
3   Send Text String "'LIST ALL^M'"
4   When Line Contains "'HOLD'"
5       Set Variable * hold from expression "theLine(CursorRow-1)"
6       Set Variable * circuit from expression "Mid(hold,9,2)"
7   End When
8   If Expression "circuit > -1"
9       Send Text String "'DISCONNECT' & circuit &'^M'"
10  End If
```

Signal Send IOCTL

Purpose: To send a control string to the current device driver.

Syntax: Signal Send IOCTL [ ]

Argument: String - in the expression box, enter a control string.

Result: The control string is sent to the device driver selected under communication settings. (See the documentation for your device driver for the syntax of the control string.) If a device driver is not currently chosen, this command is ignored.

Skip Line *

Purpose: To skip over the current line in the file named by the Send File * Text Line by Line command.

Note: Use the File * ... commands and related functions wherever possible. The Send File * Text Line by Line, Send Line * and Skip Line * commands are included only to support past versions of MicroPhone II.

Subcommands: None.

Syntax: Skip Line *

Argument: None.

Result: Skips over the current line in the file named by the Send File * Text Line by Line command and points at the following line.

A combination of the three commands - Send File * Text Line by Line, Send Line * and Skip Line * - allows you to send selected lines from a text file.

Success flag: Set to True if the line is skipped.
Set to False when the end of the file is reached.

Example: To send lines 10 through 15 from the file called newnames, use the following script:

```
1   Send File * Text Line by Line "'newnames'"
2   Set Variable * endfile From Expression "false"
3   Set Variable * counter From Expression "0"
4   Repeat
5       Skip Line *
6       If Failure
7           Set Variable * endfile From Expression "true"
8           Beep
9       End If
10      Set Variable * counter from Expression "counter + 1"
11  Until Expression "(counter = 9) OR endfile"
12  Repeat
13      Send Line *
14      If Failure
15          Set Variable * endfile From Expression "true"
16      End If
17      Set Variable * counter From Expression "counter + 1"
18  Until Expression "(counter = 15) OR endfile"
```

Trace

Purpose: To examine the operation of the script. Use this command to determine why a script does not operate properly.

Subcommands: Requires one of four subsidiary commands:

Trace On
Trace Off
Trace Single Step
Trace Report Errors

Trace On

Purpose: To activate tracing during the operation of the script.

Syntax: Trace On

Argument: None.

Result: Turns tracing on and places the trace windows on the screen showing the script command about to execute.

The first window contains the name of the script and the text of the statement that is about to execute. Subsidiary windows show the individual steps of the script command executing.

Trace Off

Purpose: To stop tracing during the operation of the script.

Syntax: Trace Off

Argument: None.

Result: MicroPhone II turns tracing off.

Trace Single Step

Purpose: To activate tracing and halt before each statement during the operation of the script.

Syntax: Trace Single Step

Argument: None.

Result: MicroPhone II turns on tracing and halts the script before executing each statement. To execute one statement, you must click the step button. That statement will be executed, and the script will halt before the next one.

Example:

```
1  Trace Single Step
2  Do Script * "'problem script'"
3  Trace Off
```

Trace Report Errors

Purpose: To report errors in script expressions.

Syntax: Trace Report Errors

Options: No or Yes - select one from the third list box.

Argument: None.

Result: If Yes is selected, MicroPhone II reports errors in expressions, whether tracing is on or not. These include such syntax errors as missing or mismatched parentheses and mismatched types in expressions. The error report will include the name of the script, the line number and a phrase describing the error.

If No is selected, MicroPhone II will ignore errors in expressions wherever possible.

Example:

```
1  Trace Report Errors Yes
2  Do Script * "'problem script'"
3  Trace Off
```

Until

Purpose: Use Until ... with the Repeat command to define the condition which, when evaluated as True, passes script execution to the statement following the Repeat construct.

The Repeat and Until ... commands enclose statements that are repeatedly executed as long as the condition defined by the Until ... command is evaluated as False. The first test is made when Until ... is first executed, so a Repeat/Until ... loop will always be executed at least once.

The Repeat construct is "static" in the sense that it does not move characters from the serial port to the terminal window. It should not be used to check for test strings as data is being received, but only when data transmission is halted. To test for new data as it streams into the terminal window, use either the Wait ... or the When ... command.

The Repeat/Until ... loop is a logical construct. See Logical Constructs.

Subcommands: Requires one of six subsidiary commands:

Until Text Equals [ ]
Until Line Contains [ ]
Until Line Does Not Contain [ ]
Until Failure
Until Success
Until Expression [ ]

Until Text Equals

Purpose: To define the condition which passes script execution to the statement following the Repeat construct when a text comparison is True.

See Until ... for more information.

Syntax: Until Text Equals [ ]

Argument: String - in the expression box, enter a test string.

Result: This test will be considered True if the text immediately to the left of the cursor on the current line is the same as the test string that is entered into the expression box.

Example:

```
1  Repeat
2      Send Text String "'^M'"
3      Wait Seconds "1"
4  Until Text Equals "'Sign in: '"
```

Until Line Contains

Purpose: To define the condition which passes script execution to the statement following the Repeat construct when a line contains the specified text.

See Until ... for more information.

Syntax: Until Line Contains [ ]

Argument: String - in the expression box, enter a test string.

Result: This test is considered True if the line above the cursor contains the test string entered into the expression box.

This test is valid when the cursor is at the far left position on a line - at column one - such as following a carriage return. Otherwise, the test will return False.

Example:

```
1  Repeat
2      Send Text String "'AT^M'"
3      Wait Seconds "2"
4  Until Line Contains "'OK'"
```

Until Line Does Not Contain

Purpose: To define the condition which passes script execution to the statement following the Repeat construct when a line does not contain the specified text.

See Until ... for more information.

Syntax: Until Line Does Not Contain [ ]

Argument: String - in the expression box, enter a test string.

Result: This test is considered True if the line above the cursor does not contain the test string entered into the expression box.

This test is valid when the cursor is at the far left position on a line - at column one - such as following a carriage return. Otherwise, the test will return False.

Example:

```
1  Repeat
2      Signal Short Break
3      Wait Seconds "1"
4  Until Line Does Not Contain "'System Busy'"
```

Until Failure

Purpose: To define the condition which passes script execution to the statement following the Repeat construct when the Success flag is False.

See Until ... for more information.

Syntax: Until Failure

Argument: None.

Result: This test will be considered True if the Success flag is evaluated as False.

Example:

```
1  Send File Text Line by Line "'MAILMSG.TXT'"
2  Repeat
3      Set Variable * index from Expression "index = index+1"
4      Set Variable * fileline from File Line
5      Send Text String "fileline"
6  Until Failure
```

Until Success

Purpose: To define the condition which passes script execution to the statement following the Repeat construct when the Success flag is True.

See Until ... for more information.

Syntax: Until Success

Argument: None.

Result: This test will be considered True if the Success flag is evaluated as True.

Example:

```
1  Repeat
2      Dial Service * "'BBS'"
3  Until Success
```

Until Expression

Purpose: To define the condition which passes script execution to the statement following the Repeat construct when an expression is True.

See Until ... for more information.

Syntax: Until Expression [ ]

Argument: Boolean - in the expression box, enter an expression that evaluates to True or False.

Result: This test will be considered True if the expression entered into the expression box is evaluated as True.

Example:

```
1   Repeat
2       Set Variable * tststrg from File Line
3       If Failure
4           Set variable * endfile from Expression "true"
5       End If
6   Until Expression "(MID(tststrg, 10, 4) = 'to: ') OR (endfile)"
```

Wait

Purpose: To pause script execution and let data stream into the terminal window until the defined condition is met.

Subcommands: Requires one of seven subsidiary commands:

Wait for Text [ ]
Wait for Line Containing [ ]
Wait for Line Not Containing [ ]
Wait Until Time [ ]
Wait Seconds [ ]
Wait Sixtieths [ ]
Wait for Button [ ]

Wait for Text

Purpose: To pause script execution and let data stream into the terminal window until the specified text is encountered.

Syntax: Wait for Text [ ]

Argument: String - in the expression box, enter a test string. The test string is usually a prompt that expects a response on the same line.

Result: MicroPhone II first tests the characters immediately to the left of the cursor and then monitors those that are received in the terminal window. When the string is encountered, or if it already appears on the screen, script execution passes to the next statement and no new data will be received.

This is a "dynamic" command. It acts as a gatekeeper looking for the test string as data is received by the program and displayed on the screen. As soon as the string is encountered, script execution passes to the next statement.

Example: Assume you are logging into a computer system that prompts "Please enter your user name:" at the beginning of a session. You may use the following command:

Wait for Text "'user name:'"

Be sure that the test string includes the last character in the prompt (in this case ":") even if it is a blank character. If not, MicroPhone II will stop transmission immediately after detecting the test string, and pass execution to the next script command before the prompt is received in its entirety. For instance, the following statement

Wait for Text "'user'"

will cause the script to halt as soon as the word "user" is encountered.

Also, you should be sure that the text you are using as the test string is unique. Otherwise, MicroPhone II will erroneously halt transmission at the first occurrence of the string, and pass execution to the next statement.

However, don't use unnecessarily long test strings. The longer the string, the more likely it is that occasional line noise will garble characters, causing valid tests to fail.

Wait for Line Containing

Purpose: To pause script execution and let data stream into the terminal window until a line containing the specified text is encountered.

Syntax: Wait for Line Containing [ ]

Argument: String - in the expression box, enter a test string.

Result: MicroPhone II first tests the line immediately above the cursor and then monitors lines that are received in the terminal window until a line containing the test string and ended by a carriage return is found. When the test string ended by a carriage return is encountered, or if it already appears on the screen, script execution passes to the next statement and no new data will be received.

This test is valid when the cursor is at the far left position on a line - at column one - such as following a carriage return. Otherwise, the test will return False.

This is a "dynamic" command. It acts as a gatekeeper, letting data stream in until it receives a full line ended by a carriage return and containing the test string from the remote. When this happens, script execution passes to the next statement.

Example: A remote may prompt you for your location by sending "Enter your city code", followed by a carriage return that places the cursor on the next line where you can enter the code. Use the command:

Wait for Line Containing "'your city'"

Unlike the Wait for Text command, in which the test string must include the last character in the prompt, here, and in the following command, you may include any text string contained in the line. The test string must be unique in the data stream sent by the remote, otherwise MicroPhone II will stop transmission at the first occurrence of the string.

Wait for Line Not Containing

Purpose: To pause script execution and let data stream into the terminal window until the first line that does not contain the specified text is encountered.

Syntax: Wait for Line Not Containing [ ]

Argument: String - in the expression box, enter a test string.

Result: MicroPhone II first tests the line immediately above the cursor and then monitors lines that are received in the terminal window until a line without the test string and ended by a carriage return is found. When a line without the test string is encountered, or if it already appears on the screen, script execution passes to the next statement and no new data will be received.

This test is valid when the cursor is at the far left position on a line - at column one - such as following a carriage return. Otherwise, the test will return False.

Example: Wait for Line Not Containing "'Working...'"

Wait Until Time

Purpose: To pause script execution and let data stream into the terminal window until the specified time.

Syntax: Wait Until Time [ ]

Argument: String - in the expression box, enter the real time in a 24-hour format, such as: 06:00 for 6 AM, or 14:10 for ten minutes after 2 PM. Time entered in this format must be enclosed in single quotes.

Result: Execution passes to the next script statement when the designated time is reached. Because this command relies on the PC's internal clock, make sure the clock is set correctly.

Example:

```
1  Wait Until Time "'17:00'"
2  Do Script * "'MCI Mail checker'"
3  Wait Until Time "'19:00'"
4  Do Script * "'CompuServe checker'"
```

Wait Seconds

Purpose: To pause script execution and let data stream into the terminal window until a number of seconds has passed.

Syntax: Wait Seconds [ ]

Argument: Integer - in the expression box, enter the number of seconds. (Remember, integers do not require single quotes.)

Result: The script will pause for the number of seconds entered into the expression box, and then pass execution to the next statement.

Example:

```
1   Repeat
2       Do Script * "'Mail Checker'"
3       Wait Seconds   "3600 { wait 1 hour }"
4   Until Expression   "HOUR = 21 { stop at 9:00 PM }"
```

Wait Sixtieths

Purpose: To pause script execution and let data stream into the terminal window until a number of sixtieths of a second has passed.

Syntax: Wait Sixtieths [ ]

Argument: Integer - in the expression box, enter the number of sixtieths of a second. (Remember, integers do not require single quotes.)

Result: The script will pause for the number of sixtieths entered into the expression box, and then pass execution to the next statement.

Example:

```
1  Send Text String "'^M'"
2  Wait Sixtieths "10"
3  Send Text String "'^M'"
```

Wait for Button

Purpose:	To pause script execution and let data stream into the terminal window until a button is clicked.
Syntax:	Wait for Button [ ]
Argument:	String - in the expression box, enter a title for the button.
Result:	The script will clear the current buttons from the bar and install the new button. It then pauses until you click that button. After the new button is clicked, it disappears, and the previous buttons are reinstalled.
Example:	Wait for Button "'Sign On'"

Wait for Call

Purpose: To set up an auto-answer modem to wait for a call.

Subcommands: None.

Syntax: Wait for Call *

Argument: None.

Result: MicroPhone II establishes a "listen" connection to the modem. It then directs the modem to answer the phone the next time it rings.

Success flag: Set to True when a call is received.
Set to False if MicroPhone II is unable to put the modem in a "wait for call" state.

Note: If the modem is successfully put in a "wait for call" state, but if no call is received, the script will never leave the Wait for Call statement.

When

Purpose: To pause script execution and let data stream into the terminal window until one of several conditions is met. When is a logical construct. See Logical Constructs.

The tests in a When construct are tried in order, from first to last. As soon as one of the tests is evaluated as True, the script executes the block of statements following that test. Then execution passes to the statement following the End When command.

If none of the tests is evaluated as True, the tests in the When construct are tried again in order. This process will repeat until one of the tests is true or the script is aborted.

Subcommands: Requires one of eight subsidiary commands:

When Text Equals [ ]
When Line Contains [ ]
When Line Does Not Contain [ ]
When Time Is [ ]
When Seconds Have Passed [ ]
When Sixtieths Have Passed [ ]
When Button [ ]
When Expression [ ]

When Text Equals

Purpose: To pause script execution and let data stream into the terminal window until a text comparison is True

See When for more information.

Syntax: When Text Equals [ ]

Argument: String - in the expression box, enter a test string.

Result: MicroPhone II first tests the characters immediately to the left of the cursor and then monitors those that are received in the terminal window. When the test string is encountered, even if no new data has been received, the block of statements that follows the When ... statement is executed.

This is a "dynamic" command. It acts as a gatekeeper looking for the test string as data is received by the program and displayed on the screen.

Example:

```
1   When Text Equals "'NO to stop: '"
2       Send Text String "'^M'"
3   Or When Text Equals "'<RETURN> to continue'"
4       Capture * Off
5       Send Text String "'^M'"
6   Or When Text Equals "'EXIT):'"
7       Send Text String "'next^M'"
8       Capture * On
9   Or When Text Equals "'Your READ request is now complete.'"
10          Send Text String "'^M'"
11  End When
```

When Line Contains

Purpose: To pause script execution and let data stream into the terminal window until a line containing the specified text is found.

See When for more information.

Syntax: When Line Contains [ ]

Argument: String - in the expression box, enter a test string.

Result: MicroPhone II first tests the line immediately above the cursor and then monitors lines that are received in the terminal window until a line containing the test string and ended by a carriage return is found.

When a line containing the test string and followed by a carriage return is encountered, or if it already appears on the screen, script execution passes to the next statement and no new data will be received.

This test is valid when the cursor is at the far left position on a line - at column one - such as following a carriage return. Otherwise, the test will return False.

This is a "dynamic" command. It acts as a gatekeeper letting data stream in until a full line ended by a carriage return and containing the test string is received from the remote.

Example:

```
1  When Line Contains "'CONNECT'"
2      Do Script * "'Sign In'"
3  Or When Line Contains "'NO CARRIER'"
4      Return Failure
5  End When
```

When Line Does Not Contain

Purpose: To pause script execution and let data stream into the terminal window until the first line without the specified text is found.

See When for more information.

Syntax: When Line Does Not Contain [ ]

Argument: String - in the expression box, enter a test string.

Result: MicroPhone II first tests the line immediately above the cursor and then monitors lines that are received in the terminal window until a line without the test string and ended by a carriage return is found.

When a line without the test string and ended by a carriage return is encountered, or if it already appears on the screen, script execution passes to the next statement and no new data will be received.

This test is valid when the cursor is at the far left position on a line - at column one - such as following a carriage return. Otherwise, the test will return False.

Example:

```
1  When Line Does Not Contain "':'"
2      Send Text String "'exit'"
3  Or When Text Equals "'File Empty'"
4      Chain to Script "'Initialize'"
5  End When
```

When Time Is

Purpose: To pause script execution and let data stream into the terminal window until the time specified.

See When for more information.

Syntax: When Time Is [ ]

Argument: String - in the expression box, enter real time in a 24-hour format, such as: 02:10 for ten minutes after 2 AM, or 14:10 for ten minutes after 2 PM. Time entered in this format must be enclosed in single quotes.

Result: This test will be considered True when the System clock registers the indicated time.

Example:

```
1   When Time Is "'8:00'"
2       Do Script * "'Daily Mail'"
3   Or When Button "'Cancel'"
4       Abort Script
5   End When
```

When Seconds Have Passed

Purpose: To pause script execution and let data stream into the terminal window until a number of seconds has passed.

See When for more information.

Syntax: When Seconds Have Passed [ ]

Argument: Integer - in the expression box, enter the number of seconds. (Remember, integers do not require single quotes.)

Result: This test will be considered True when the indicated number of seconds have passed. Time is measured from the instant the When construct is first encountered.

Example:

```
1  When Seconds Have Passed "30"
2      Chain to Script "'Log Off'"
3  Or When Text Equals "'Next?'"
4      Send Text String "'Check Account^M'"
5  End When
```

When Sixtieths Have Passed

Purpose: To pause script execution and let data stream into the terminal window until a number of sixtieths of a second has passed.

See When for more information.

Syntax: When Sixtieths Have Passed [ ]

Argument: Integer - in the expression box, enter the number of sixtieths of a second. (Remember, integers do not require single quotes.)

Result: This test will be considered True when the indicated number of sixtieths have passed. Time is measured from the instant the When construct is first encountered.

Example:

```
1  When Sixtieths Have Passed "80"
2      Chain to Script "'Shut Down'"
3  Or When Expression "NextKey = 'A'"
4      Do Script * "'Reset'"
5  End When
```

When Button

Purpose: To pause script execution and let data stream into the terminal window until a button is clicked.

See When for more information.

Syntax: When Button [ ]

Argument: String - in the expression box, enter a title for the button.

Result: MicroPhone II will temporarily clear the current buttons and display a new button on the bar. The test will be considered True if you click that button. Once clicked, the button disappears and the previous buttons are reinstalled.

Subsequent When Button statements in the same When clause add buttons without clearing the button bar each time. Buttons are installed while the script runs.

Example:

```
1  When Button "'Send a message'"
2      Send Text String "'send'"
3  Or When Seconds Have Passed "20"
4      Send Text String "'exit'"
5  End When
6  Send Text String "'^M'"
```

Note: Multiple buttons may be installed so that, during script execution, you may choose one of several options by clicking on a button. Install multiple buttons using Or When statements.

When Expression

Purpose: To pause script execution and let data stream into the terminal window until the expression is True.

See When for more information.

Syntax: When Expression [ ]

Argument: Boolean - in the expression box, enter an expression that evaluates to True or False.

Result: This test is considered True if the expression entered into the expression box is evaluated as True.

Example:

```
1  When Expression "MID(testline,10,8) = 'accepted'"
2        Do Script * "'Next Message'"
3  Or When Expression "POS(testline, 'Addressee not found') > 1"
4        Do Script * "'Get new address'"
5  End When
```

While

Purpose: To execute a block of statements as long as a defined condition is met.

The While construct is "static" in the sense that it does not move characters from the serial port to the terminal window. It should not be used to check for test strings as data is being received, but only once data transmission is halted. To test dynamic conditions, such as new data as it streams into the terminal window, use either the Wait ... or When commands.

While ... is a logical construct. See Logical Constructs.

The While ... and End While commands enclose statements that are repeatedly executed as long as the condition defined by the While ... command is evaluated as True. The first test of this statement is made when While ... is first executed; thus, a While/End While loop may never be executed if the condition it defines is never True.

The End While command closes the group of statements within the While ... construct.

Subcommands: Requires one of six subsidiary commands:

While Text Equals [ ]
While Line Contains [ ]
While Line Does Not Contain [ ]
While Failure
While Success
While Expression [ ]

While Text Equals

Purpose: To execute a block of statements as long as a text comparison is True.

See While... for more information.

Syntax: While Text Equals [ ]

Argument: String - in the expression box, enter a test string.

Result: This test is considered True if the text immediately to the left of the cursor on the current line is the same as the test string that is entered into the expression box.

Example:

```
1   While Text Equals "'>'"
2       Send Text String "'^M'"
3       Wait Seconds "1"
4   End While
```

While Line Contains

Purpose: To execute a block of statements as long as a line containing the specified text is found.

See While... for more information.

Syntax: While Line Contains [ ]

Argument: String - in the expression box, enter a test string.

Result: This test is considered True if the line above the cursor contains the test string entered into the expression box.

This test is valid when the cursor is at the far left position on a line - at column one - such as following a carriage return. Otherwise, the test will return False.

Example:

```
1   While Line Contains "'..."
2       Send Local to Screen "NextCh"
3   End While
```

While Line Does Not Contain

Purpose: To execute a block of statements as long as a line not containing the specified text is found.

See While... for more information.

Syntax: While Line Does Not Contain [ ]

Argument: String - in the expression box, enter a test string.

Result: This test is considered True if the line above the cursor does not contain the test string entered into the expression box.

This test is valid when the cursor is at the far left position on a line - at column one - such as following a carriage return. Otherwise, the test will return False.

Example:

```
1  While Line Does Not Contain "'System Ready'"
2      Signal Flash DTR
3      Wait Seconds "3"
4  End While
```

While Failure

Purpose: To execute a block of statements as long as the Success flag is False.

See While... for more information.

Syntax: While Failure

Argument: None.

Result: This test will be considered True if the Success flag is evaluated as False.

Example:

```
1   While Failure
2       Dial Service * "'BBS'"
3   End While
```

While Success

Purpose: To execute a block of statements as long as the Success flag is True.

See While... for more information.

Syntax: While Success

Argument: None.

Result: This test will be considered True if the Success flag is evaluated as True.

Example:

```
1   While Success
2       Do Script * "'Next Message'"
3   End While
```

While Expression

Purpose: To execute a block of statements as long as an expression is True.

See While... for more information.

Syntax: While Expression [ ]

Argument: Boolean - in the expression box, enter an expression that evaluates to True or False.

Result: This test is considered True if the expression entered into the expression box is evaluated as True.

Example: You may want to Beep repeatedly. To Beep 15 times, use the following statements:

```
1   Set Variable * NumTries from Expression "0"
2   While Expression "NumTries < 15"
3       Beep
4       Set Variable * Numtries from Expression "NumTries + 1"
5   End While
```

Operator Precedence

Generally, operators are used between the values on which they operate. For example:

A * B

There are two exceptions. NOT is a "unary" operator that acts only on the value after it. Subtraction (-) may be used as a unary operator, - (a * 56) for example.

The rules below define the precedence of operators when more than one operator appears in an expression. The operators are grouped in order of precedence. An operator has precedence over those below it. Those on the same line have the same precedence.

Highest precedence	( )
	NOT
	* / AND DIV MOD XOR
	+ - OR
	= <> >= <= > <
Lowest precedence	&

The first rule:
Where a choice is possible, the operator with higher precedence will be executed first. In the example

A + B * C

B and C will be multiplied first, then A will be added to the result. NOT A > B will be evaluated by taking the logical NOT of A and then testing against the value of B.

The second rule:
Operators with the same precedence are evaluated from left to right. For example:

A * B DIV C

is evaluated by multiplying A and B, then dividing by C.

The third rule:
Parentheses may be used to enforce a precedence that is contrary to the first two rules. For example:

(A + B) * C forces the calculation of A + B first, followed by * C.

NOT (A > B) forces the comparison first, then the logical NOT.

AND

Values must be of boolean or integer type. If the values are boolean, the result is the logical AND of the two values (TRUE or FALSE). If the values are integers, the result is an integer that is the bitwise AND of the values.

Examples:	(3 < 5) AND (5 > 8)	= FALSE
	3 AND 5	= 1
	3 AND 0	= 0

DIV

The values must be of integer type. The result is the quotient of the left value divided by the right. The remainder is discarded. Division by 0 is an error.

Examples:

3 DIV 5 = 0

5 DIV 3 = 1

5 DIV 0 = error (returns 5)

MOD

The values must be of integer type. The result is the remainder after integer division of the left value by the right. The quotient is discarded. Mod by 0 is an error.

Examples:	3 MOD 5	=	3
	5 MOD 3	=	2
	60 MOD 0	=	error (returns 60)

NOT

NOT is a unary operator. It only affects the value to its right. The value must be boolean or integer type. If boolean, the result is the logical NOT of the value. (If integer, the result is the bitwise inverse of the value.)

Examples: NOT (3 < 5) = FALSE

NOT 3 = -4

OR

The values must be of boolean or integer type. If boolean, the result is the logical OR of the values. (If integer, it is the bitwise OR of the integers.)

Examples: (3 < 5) OR false = TRUE

3 OR 5 = 7

XOR

The values must be of boolean or integer type. If boolean, the result is the logical XOR of the values. (If integer, it is the bitwise XOR of the integers.)

Examples:

```
(3 < 5) XOR false   =   TRUE

(3 < 5) XOR true    =   FALSE

      3 XOR 5       =   6
```

*** (Multiplication)**

The values must be of integer type. The result is the product of the two values.

Example: 3 * 5 = 15

/ (Division)

The values must be of integer type. The result is the integer division of the two values; / is identical in effect to DIV

- (Subtraction)

The values must be of integer type. The result is the difference of the two values.

Example: 3 - 5 = -2

+ (Addition)

The values must be of integer type, and the result is the sum of the two values.

Example: 3 + 5 = 8

= > < <= >= <> (Comparison Operators)

These are comparison operators that work with all types of values. If the types are not the same, they will both be converted to strings. The result is of boolean type and indicates whether the relation is True or False.

Examples:

3 < 5	=	TRUE
true >= false	=	TRUE
'and' > 'andy'	=	FALSE

& (Ampersand)

The ampersand is for concatenation of strings. All types of values are treated as character strings. The result is a string constant made of the two values run together in the order that they appeared.

Examples:	'abc' & 'def'	=	'abcdef'
	'abc' & 34	=	'abc34'
	'abc' & false	=	'abcfalse'
	true & 34	=	'true34'

ApplicationPath

Purpose: To obtain the pathname to MicroPhone II.

Syntax: ApplicationPath

Argument: This function takes no argument.

Returns: A string specifying the pathname to MicroPhone II, including drive specifier.

ASCII(s)

Purpose: To determine the ASCII value that corresponds to a character.

Syntax: ASCII(string)

Argument: String - A group of characters, or an expression, evaluated as a string.

Returns: An integer that is the ASCII value of the first character in the string.

Examples:

```
ASCII('abcdef')                          97
ASCII(3<5) {argument is TRUE}      84
```

Notes: If the argument is not of string type, it will be converted to a string.

BaudRate

Purpose: To determine the current baud rate communications setting.

Syntax: BaudRate

Argument: This function takes no argument.

Returns: A string representing the value of the baud rate setting in the Communications Settings dialog box.

Example: BaudRate 2400

BitsPerChar

Purpose: To determine the current bits per character communications setting.

Syntax: BitsPerChar

Argument: This function takes no argument.

Returns: A string representing the value of the bits per character (also known as data bits) setting in the Communications Settings dialog box.

Example: BitsPerChar 7

Char(i)

Purpose:	To determine the character that corresponds to an ASCII value.
Syntax:	Char(value)
Argument:	Integer - A number that represents a member of the ASCII character set.
Returns:	A string of length 1 whose first (and only) character corresponds to the integer's ASCII code.
Example:	Char(116) t
Note:	If the argument is not an integer, it will be converted to an integer.

ConnectionDriver

Purpose: To obtain the name of the current connection driver.

Syntax: ConnectionDriver

Argument: This function takes no argument.

Returns: A string specifying the name of the current connection driver. This identifies the document that provides MicroPhone II instructions on how to establish a connection through the chosen connection port.

ConnectionOpen

Purpose: To determine if the current connection is open.

Syntax: ConnectionOpen

Argument: This function takes no argument.

Returns: A boolean value; TRUE if the connection specified in the Communications Settings dialog box (in the Settings menu) is currently open. This setting is controlled by the Connection Open and Connection Close commands.

ConnectionParameter

Purpose: To obtain the configuration settings of the current connection port.

Syntax: ConnectionParameter

Argument: This function takes no argument.

Returns: A comma delimited string specifying the parameter settings used to configure the current connection port. The Item functions may be used to separate the items in the list. The comma is omitted if there is only one parameter.

Example If the current settings document is set to use a NetBIOS connection, ConnectionParameter will return a value similar to this:

0, "Panda", "London, ""et. al.""", 3, 2, 1

Each item represents a setting in MicroPhone II's Communications Settings dialog box. The first item, 0, represents the Client Mode. "Panda" corresponds to entering the string Panda under Local Name in MicroPhone II's Communications Settings dialog box (when NetBIOS is the selected Connection).

"London, ""et. al.""" corresponds to the entry London, "et. al." for the Remote Name. The local and remote names are within double quotes because NetBIOS permits the names to contain any printable characters. If the string contains a double quote, then the double quote must be preceded by another double quote.

The fourth item, 3, is the Receive Time Out value. The fifth item, 2, is the Send Time Out. The last item, 1, indicates Blank Padding is used. (0 would indicate Blank Padding turned off.)

ConnectionPath

Purpose: To obtain the pathname of the current connection driver.

Syntax: ConnectionPath

Argument: This function takes no argument.

Returns: A string specifying the path of the current connection driver, including drive specifier.

ConnectionPort

Purpose: To obtain the name of the current connection port.

Syntax: ConnectionPort

Argument: This function takes no argument.

Returns: A string specifying the name of the current connection port. This identifies the hardware connection used to connect to a modem or remote computer. (The connection port is normally defined by the connection driver.)

CursorCol

Purpose:	Used with CursorRow, to determine the position of the cursor in the terminal window.
Syntax:	CursorCol
Argument:	This function takes no argument.
Returns:	An integer that is the column number of the current screen cursor position.
Note:	The column number will be between 1 and the value chosen under Communications Settings.

CursorRow

Purpose:	Used with CursorCol to determine the position of the cursor in the terminal window.
Syntax:	CursorRow
Argument:	This function takes no argument.
Returns:	An integer that is the row number of the current screen cursor position.
Note:	The row number will be between 1 and the value chosen under Communications Settings.

Day

Purpose: To obtain the current day of the month from the System date.

Syntax: Day

Argument: This function takes no argument.

Returns: An integer that is the current day of the month, from 1 to 31.

DayOfWeek

Purpose: To obtain the current day of the week from the System date.

Syntax: DayOfWeek

Argument: This function takes no argument.

Returns: An integer that is the current day of the week, from 1 to 7 (Sunday to Saturday).

EOF(s)

Purpose: To determine if the filemark is at the end of a file.

Syntax: EOF(filename)

Argument: Filename - A string, or an expression evaluated as a string, that is the name of a file opened during the session with the File*Open script command.

Returns: A boolean that is TRUE if the filemark is at the end of the file, and FALSE if the filemark is not at the end of the file.

Notes: If the file is not open, the function will return TRUE.

EOLN(s)

Purpose: To determine if the filemark is at the end of a line.

Syntax: EOLN(filename)

Argument: Filename - A string, or an expression evaluated as a string, that is the name of a file opened during the session with the File*Open script command.

Returns: A boolean that is TRUE if the filemark is at the end of a line, and FALSE if the filemark is not at the end of a line.

Notes: A carriage return is considered the end of a line.

The function returns TRUE if the filemark is on a blank line. It will also return TRUE if the filemark has reached the end of the file.

If the file is not open, the function will return TRUE.

Exists(v)

Purpose: To determine if a variable has been established.

Syntax: Exists(variablename)

Argument: Variablename - The name of a variable.

Returns: A boolean that is TRUE if the variable exists and FALSE if it does not.

Exists(ach) is TRUE if the variable ach has been used in the current MicroPhone II session and was not been deleted since its use.

Note: A variable exists after it has been assigned a value by the Set Variable * ... script command.

FileChooseOne(s,s)

Purpose: To allow the user to choose one item from a list.

Syntax: FileChooseOne(filename, prompt)

Arguments: Filename - A text file containing a list item on each line.

Prompt - A string containing a prompt that will appear in the choose dialog.

Returns: The index of the item selected in the dialog box.

Notes: The function presents the user with a dialog box containing a list of the values in the file indicated in the first argument. The box also contains the reminder prompt (the second argument). Double-clicking an item is the same as selecting the item and choosing the OK button. Script execution is suspended while the dialog box is on the screen.

Clicking the Cancel button in this dialog box returns zero (FALSE).

FileLen(s)

Purpose: To determine the size of a file.

Syntax: FileLen(filename)

Argument: Filename - A string, or an expression evaluated as a string, that is the name of any file.

Returns: An integer that is the length of the file in bytes. (This is not necessarily the same as the number of bytes from the beginning of the file to the end-of-file mark. Use the GetEOF function to obtain that value.)

Note: The FileLen function is similar to the GetEOF function. However, to maintain cross-platform compatibility, GetEOF should be used to get the length of a file whose contents are changed by MicroPhone II scripts. FileLen should be used for files whose contents are closed or not changed by MicroPhone II scripts.

FileLine(s)

Purpose: To determine the line number on which the filemark is positioned in a file.

Syntax: FileLine(filename)

Argument: Filename - A string, or an expression evaluated as a string, that is the name of a file opened during the session with the File*Open script command.

Returns: An integer that is the number of the line on which the filemark is located.

Note: Numbering begins with 0. (The tenth line of the file is number 9.)

FilePos(s)

Purpose: To determine the position of the filemark in a file.

Syntax: FilePos(filename)

Argument: Filename - A string, or an expression evaluated as a string, that is the name of a file opened during the session with the File*Open script command.

Returns: An integer that is the number of bytes the filemark is offset from the beginning of the file.

Note: Numbering begins with 0.

FillStr(s,i)

Purpose: To generate a string of repeating characters.

Syntax: FillStr(string, count)

Arguments: String - A group of characters, or an expression evaluated as a string.

Count - An integer that is the number of times string will be repeated.

Returns: A string filled with string, repeated count times.

Example: FillStr('Yes',3) 'YesYesYes'

GetEOF(s)

Purpose: To determine the end-of-file mark from the beginning of a file.

Syntax: GetEOF(filename)

Argument: Filename - A string, or an expression evaluated as a string, that is the name of a file opened during the session with the File*Open script command.

Returns: An integer that is the number of bytes from the beginning of the file to the end-of-file mark. (This is not necessarily the same as the length of the entire file. Use the FileLen function to obtain that value.)

Note: The GetEOF function is similar to the FileLen function. However, to maintain cross-platform compatibility, GetEOF should be used to get the length of a file whose contents are changed by MicroPhone II scripts. FileLen should be used for files whose contents are closed or not changed by MicroPhone II scripts.

Hour

Purpose:	To obtain the current time in hours from the System clock.
Syntax:	Hour
Argument:	This function takes no argument.
Returns:	An integer representing the hour of the day, from 0 to 23.

Insert(s,s,i)

Purpose: To insert one string within another.

Syntax: Insert(sourcestr, deststr, startloc)

Arguments: Sourcestr - A group of characters or an expression evaluated as a string to be inserted into the string specified by Argument 2.

Deststr - Another group of characters or an expression evaluated as a string into which the source string is inserted.

Startloc - A integer that represents the character position in the destination string at which the source string will be inserted.

Returns: A string in which the source string is inserted in the destination string, beginning at the start location within the destination string.

Example: Insert('arf', 'Sandy', 4) 'Sanarfdy'

ItemCount(s,s)

Purpose: To determine the number of items in a string. Items are defined as text strings separated from each other by a single character delimiter.

Syntax: ItemCount(itemList, delimiter)

Arguments: ItemList - a string containing a number of items.

Delimiter - a string whose first character is the delimiter used to separate items in itemList.

Returns: The number of items in itemList.

Note: Empty items (null strings) are allowed. It is assumed that there is a string before and after each delimiter. If there are two delimiters together, ItemCount assumes there is a null string between them. If there is a delimiter at the beginning or end of itemList, ItemCount assumes there is a null string before or after the delimiter, respectively.

Examples:

```
1  Set Variable count From Expression "ItemCount('a:b:c:d', ':')"
2  Remark "count becomes 4"
```

```
1  Set Variable count From Expression "ItemCount('a:b:c:d::f', ':')"
2  Remark "count becomes 6"
```

ItemDelete(s,s,i)

Purpose: To delete an item from a string containing a list of items.

Syntax: ItemDelete(itemList, delimiter, targetItem)

Arguments: ItemList - a string containing a number of items.

Delimiter - a string whose first character is the delimiter used to separate items in the itemList.

TargetItem - a number indicating which item to delete.

Returns: A string consisting of itemList without the item that was deleted. If the item does not exist, then the original itemList is returned.

Note: Empty items (null strings) are allowed. See ItemCount for an explanation of how items are counted.

Examples:

```
1  Set Variable newList From Expression "ItemDelete('a:b:c:d' , ':', '3')"
2  Remark "newList returns 'a:b:d'"
```

```
1  Set Variable newList From Expression "ItemDelete('a:b:c:d' , ':', '7')"
2  Remark "newList returns 'a:b:c:d' because there are not 7 items in the list"
```

ItemFetch(s,s,i)

Purpose: To extract a specific item from a string containing a list of items.

Syntax: ItemFetch(itemList, delimiter, targetItem)

Arguments: ItemList - a string of a number of delimited items.

Delimiter - a string whose first character is the delimiter used to separate items in the itemList.

TargetItem - a number indicating the item to extract.

Returns: A string containing only the item that was specified. If the item does not exist, then an empty string is returned.

Note: Empty items (null strings) are allowed. See ItemCount for an explanation of how items are counted.

Examples:

```
1  Set Variable theItem From Expression "ItemFetch('a:b:c:d' , ':', '3')"
2  Remark "theItem returns 'c'."
```

```
1  Set Variable theItem From Expression "ItemFetch('a:b:c:d' , ':', '7')"
2  Remark "theItem returns " (a pair of single quotes represents a null string) because there is not a seventh item in the list."
```

ItemInsert(s,s,s,i)

Purpose: To insert a specific item into a string containing a list of items.

Syntax: ItemInsert(insertItem, itemList, delimiter, afterItem)

Arguments: InsertItem - a string containing the item to be inserted.

ItemList - a string of a number of delimited items.

Delimiter - a string whose first character is the delimiter used to separate items in the itemList.

AfterItem - an integer that represents the item after which insertItem will be inserted.

Returns: A string containing the item list with the inserted item.

Note: Empty items (null strings) are allowed. See ItemCount for an explanation of how items are counted.

Examples:

```
1  Set Variable newList From Expression "ItemInsert('x', 'a:b:c:d' , ':', '3')"
2  Remark "newList returns 'a:b:c:x:d'"
```

```
1  Set Variable newList From Expression "ItemInsert('x', 'a:b:c:d' , ':', '0')"
2  Remark "newList returns 'x:a:b:c:d' because the item was inserted after a nonexistent item 0, and before item 1."
```

```
1  Set Variable newList From Expression "ItemInsert('x', 'a:b:c:d' , ':', '7')"
2  Remark "newList returns 'a:b:c:d:::x'. This happens because the original list only contained 4 items, thus three empty items (item 5, 6, and 7) had to be created so that the item could be inserted after item 7."
```

ItemReplace(s,s,s,i)

Purpose: To replace a specific item in a string containing a list of items.

Syntax: ItemReplace(newItem, itemList, delimiter, targetItem)

Arguments: NewItem - the new item (text string) to replace the target item.

ItemList - a string containing a number of delimited items into which the newItem is placed.

Delimiter - a string whose first character is the delimiter used to separate items in the itemList.

TargetItem - an integer that represents the position in the itemList of the item to be replaced.

Returns: A string containing the item list with the replaced item.

Note: Empty items (null strings) are allowed. See ItemCountfor an explanation of how items are counted.

Examples:

```
1  Set Variable newList From Expression "ItemReplace('x' , 'a:b:c:d' , ':', '3')"
2  Remark "newList returns 'a:b:x:d'"
```

```
1  Set Variable newList From Expression "ItemReplace('x' , 'a:b:c:d' , ':', '0') "
2  Remark "newList returns 'a:b:c:d' because there is no item 0 to replace"
```

```
1  Set Variable newList From Expression "ItemReplace('x' , 'a:b:c:d' , ':', '7') "
2  Remark "newList returns 'a:b:c:d' because there is no item 7 to replace"
```

Length(s)

Purpose:	To determine the length of a string.	
Syntax:	Length(string)	
Argument:	String - A group of characters (or an expression) evaluated as a string.	
Returns:	An integer equal to the number of characters in the string.	
Examples:	Length('abcdef')	6
	Length(87)	2
Note:	If the argument is not of string type, it will be converted as though the String function had been applied.	

Max(e,e...)

Purpose: To determine the greatest of a group of values.

Syntax: Max(value, value ...)

Arguments: Value, value ... - Two or more values (string, integer or boolean) that are all of the same type.

Returns: The greatest value from the group.

Examples: Max(1,2,3,4,5) 5

Max('an', 'and') and

Note: If the values are of different types, they will be changed to strings.

Mid(s,i,i)

Purpose: To extract one or more characters from a string.

Syntax: Mid(string, startloc, length)

Arguments: String - A group of characters (or an expression) evaluated as a string.

Startloc - An integer that represents the character position in the string where the extraction will begin.

Length - An integer that is the length of the substring to be extracted.

Returns: A string, of Length number of characters, that contains the extracted characters from the argument String, beginning at position Startloc.

Example: Mid('funny', 3, 2) 'nn'

Notes: Mid and Pos can be used to implement a translation table. For example, assuming ch is a string variable of length 1:

Set Variable ch from Expression "Mid('abc', Pos(ch, 'xyz'), 1)"

will change the value of variable ch in this way: 'x' to 'a', 'y' to 'b' and 'z' to 'c'. Any other character is changed to the null string.

Min(e,e...)

Purpose: To determine the smallest of a group of values.

Syntax: Min(value, value ...)

Arguments: Value, value ... - Two or more values (string, integer or boolean) that are all of the same type.

Returns: The smallest value from the group.

Examples: Min(1,2,3,4,5) 1

Min('an', 'and') an

Note: If the values are of different types, they will be changed to strings.

Minute

Purpose:	To obtain the current time in minutes from the System clock.
Syntax:	Minute
Argument:	This function takes no argument.
Returns:	An integer representing the minute of the current hour, from 0 to 59.

ModemDriver

Purpose: To obtain the name of the current modem driver.

Syntax: ModemDriver

Argument: This function takes no argument.

Returns: A string specifying the name of the current modem driver. This identifies the document used to instruct the modem how to connect or disconnect from a telephone connection.

ModemPath

Purpose: To obtain the pathname of the current modem driver.

Syntax: ModemPath

Argument: This function takes no argument.

Returns: A string specifying the path to the current modem driver, including drive specifier.

Month

Purpose: To obtain the current month from the System date.

Syntax: Month

Argument: This function takes no argument.

Returns: An integer representing the current month of the year, from 1 to 12.

NextCh

Purpose: To extract a character from the serial port before MicroPhone II processes it.

Syntax: NextCh

Argument: This function takes no argument.

Returns: A string of length 1 with the latest character from the serial port.

If there is no character currently available, the function returns a string of length 0.

Notes: This function bypasses the rest of MicroPhone II. The character is not altered or written to the screen. To do that, you must use the Send Local to Screen script command.

For example, when receiving text to a file (which shows the characters on the screen before saving them), many non-printing or control characters, such as form feeds, will be converted or removed. With NextCh, a control character can be taken from the serial connection, saved without change (with the Send Local to File script command) and then displayed on the screen (if desired). Screen appearance will be the same as before, but the file contents will now include form feeds. All original control characters can be preserved.

Use NextCh to customize your screen display. Apply this function to filter characters before they are sent to the screen, and then draw the screen to your specifications with the use of screen addressing and Send Local to Screen script commands.

NextKey

Purpose: To access characters typed from your keyboard.

Syntax: NextKey

Argument: This function takes no argument.

Returns: A string of length 1 with the latest character typed at the keyboard.

Notes: If there is no character currently available, the function returns a string of length 0.

The NextKey function allows characters to be typed while a script is running.

NextKey can be used to process characters before displaying them or saving them. The character is not altered or written to the screen. If you want it there, you must use the Send Local to Screen script command.

Number(s)

Purpose: To convert a string of digits to its integer value.

Syntax: Number(string)

Argument: String - A group of characters, or an expression, evaluated as a string.

Returns: An integer.

Example: Number('87') 87

Notes: A boolean is changed to 0 if False, 1 if True. A string of digits is converted to the decimal number it represents. A string containing characters other than digits, or a leading + or -, will be converted to a number, but the value will be essentially meaningless.

Note that Number "reads" the entire string and converts it into an integer. The ASCII function takes the first character of the string and returns its integer code.

Parity

Purpose: To determine the current value of the parity communications setting.

Syntax: Parity

Argument: This function takes no argument.

Returns: A string representing the current value of the parity setting: none, even, odd, space or mark.

Pos(s,s)

Purpose: To determine if, and where, a character sequence may be found within a string.

Syntax: Pos(searchStr, sourceStr)

Arguments: searchStr - A group of characters, or an expression, evaluated as a string.

sourceStr - Another group of characters, or an expression, evaluated as a string.

Returns: If the search string is found in the source string, the result is the position of the first character of the search string within the source string. The result is 0 if the search string is not contained in the source string.

Examples:

```
Pos('a','now is the time')          0

Pos('a', 'now is a good time')      8
```

Notes: If either of the arguments is not a string, it will be converted to a string.

Pos can be used to test a character to see if it is a member of a set of characters. For example, Pos(ch, '1234567890') is not 0 if ch is a digit; 0, if it is any other character.

ReadCh(s)

Purpose: To obtain the character in a file to which filemark is pointing.

Syntax: ReadCh(filename)

Argument: Filename - A string, or an expression evaluated as a string, that is the name of a file opened during the session with the File*Open script command.

Returns: A string (of length one) that is the character pointed to by the filemark.

Note: After this function is performed, the filemark is advanced to the next character.

ReadLn(s)

Purpose: To obtain the characters from the filemark to the end of the line.

Syntax: ReadLn(filename)

Argument: Filename - A string, or an expression evaluated as a string, that is the name of a file opened during the session with the File*Open script command.

Returns: A string, containing zero to 255 characters, from the filemark to the next carriage return, but not the carriage return itself.

Note: If there are more than 255 such characters, then the first 255 characters are returned.

After this function is performed, the filemark is advanced to the beginning of the next line (the character directly after the carriage return).

ReadStr(s,i)

Purpose: To obtain a specified number of characters from the filemark.

Syntax: ReadStr(filename, length)

Argument: Filename - A string, or an expression evaluated as a string, that is the name of a file opened during the session with the File*Open script command.

Length - A number, or an expression evaluated as a number, that is the number of characters to be read.

Returns: A string, containing zero to 255 characters, from the filemark to length number of characters past the filemark.

Note: If more than 255 characters are specified, then the first 255 characters are returned.

If the end-of-file is reached before the specified number of characters are read, only the characters up to the end-of-file are returned.

After this function is performed, the filemark is advanced to the next unread character.

ReceivePath

Purpose: To obtain the pathname of the current Receive Directory.

Syntax: ReceivePath

Argument: This function takes no argument.

Returns: A string specifying the path to the current Receive Directory, including drive specifier. This is where a file received during a MicroPhone II file transfer will be saved.

Second

Purpose: To obtain the current time in seconds from the System clock.

Syntax: Second

Argument: This function takes no argument.

Returns: An integer representing the current second of the minute, from 0 to 59.

Example: Second 58

SettingsPath

Purpose: To obtain the pathname of the directory containing the current settings document.

Syntax: SettingsPath

Argument: This function takes no argument.

Returns: A string specifying the path to the directory of the current settings document, including drive specifier.

StopBits

Purpose: To determine the current value of the stop bits communications setting.

Syntax: StopBits

Argument: This function takes no argument.

Returns: A string representing the current value of the stop bits setting: auto, 1, 1.5 or 2.

String(e)

Purpose: To convert an expression of any type to a string.

Syntax: String(value)

Arguments: Value - A string, integer or boolean.

Returns: A string.

Examples:

String('abcdef')	'abcdef'
String(3<5)	'TRUE'
String(57-35)	'22'

Note: Integers are converted to a string representing the value in decimal notation. Booleans are converted to 'FALSE' or 'TRUE'. Strings are unchanged.

Success

Purpose: To ascertain the current status of the Success.

Syntax: Success

Argument: This function takes no argument.

Returns: A boolean, TRUE or FALSE, based upon the current value of MicroPhone II's Success flag.

Example: This function can be especially useful with logical constructs, such as in a script statement like this:

If Expression "(a = b) OR Success"

TheLine(i)

Purpose: To obtain a line of data from the screen.

Syntax: TheLine(linenum)

Argument: Linenum - A number from 1 to the Column terminal setting indicating a line on the screen.

Returns: A string that contains the contents of the screen line specified.

Note: The line in which the cursor is currently located can be accessed by using the function TheLine(CursorRow).

TickCount

Purpose:	To obtain the current tick count.
Syntax:	TickCount
Argument:	This function takes no argument.
Returns:	An integer whose value is the current tick count from the system.
Note:	The tick count begins at zero when the system starts up and increases by one every sixtieth of a second.

TypeOf(e)

Purpose: To ascertain the variable type of a value.

Syntax: TypeOf(value)

Arguments: Value - A string, integer or boolean.

Returns: A string which is the name of the type of the expression: 'string', 'integer' or 'boolean'.

Examples:

TypeOf('abcdef')	'string'
TypeOf(87)	'integer'
TypeOf(3<5)	'boolean'

UCase(s)

Purpose: To convert the characters in a string to upper case.

Syntax: UCase(string)

Arguments: String - A group of characters, or an expression, evaluated as a string.

Returns: The same string with all characters converted to upper case.

Example: UCase('abcdef') 'ABCDEF'

UCase(3<5) 'TRUE'

Note: If the argument is not of type string, it will be implicitly converted to a string.

Year

Purpose:	To obtain the current year from the System date.
Syntax:	Year
Argument:	This function takes no argument.
Returns:	A two-digit integer representing the current year.
Example:	Year 89

Elements of an Expression

An expression may be constructed from the following elements:

String constants	'store' 'C:\CommFldr\CapFile.txt'
Integer constants	27
Boolean constants	TRUE FALSE
Variable names	my_var filenames
Operators -	
Arithmetic	+ - * / DIV MOD
Comparison	= <> < > <= >=
Logical	AND OR XOR NOT
String Concatenation	&
Function names	Pos Day
Delimiters	( ) ,

A **string constant** may be any group of up to 253 characters and must be enclosed in single quotation marks so that it won't be interpreted as a variable. If a string constant contains an apostrophe, it must be written as two single quotes, such as 'don''t'.

An **integer constant** is composed of digits and may include a negative sign (-). (When MicroPhone II converts integer values to boolean values, zero is interpreted as FALSE and all other values are TRUE.)

The **boolean constants** are TRUE and FALSE. Upper or lower case letters may be used.

A variable may contain up to 24 characters. It should begin with an alphabetical character but may contain digits after the initial character.

Operators are special symbols that are used between two values and compute a result based on those values.

MicroPhone II's functions are listed in the function reminder box in the Script Editor. Function names are not case sensitive.

MicroPhone II has two delimiters: parentheses and commas. Parentheses are used to distinguish a function's name from its argument and to enclose parts of expressions. A comma is used to separate arguments.

Expression Syntax

In general, syntax rules for MicroPhone II are very similar to those found in most common programming languages, especially Pascal. For example,

xf + 3

follows correct syntax rules. A + sign always occurs between the values to be added, and spaces may be used to separate items for readability.

Errors in expressions can occur because the syntax is wrong, an operation was attempted on a value of the wrong type or a value is bad. Some examples:

34 56 + is bad syntax: the + operator must be between the values.

'abc' + 'def ' is wrong because + applies to integers, not strings.

ffd DIV x will be wrong if x is 0.

All errors will be reported to the user in an alert box if the **Report errors in expressions** choice is made in the Trace options dialog box (accessible from the Scripts Menu). If this option is not selected, you will not be alerted, and MicroPhone II will try to do the most sensible thing with the results.

For example, 3 & 67 is a type error because the & operator expects the values to be strings instead of integers. This operation will convert the integers to strings, concatenate them and continue (whether or not **Report errors in expressions** is on).

Comments may be embedded within an expression by using braces { }. MicroPhone II skips over the braces and everything between them (unless, of course, they are a part of a string).

Variables

MicroPhone II's script language can work with variables of three types: integer, string and boolean.

> An **integer** is stored in a single, 4-byte (32 bit) unit, called a word and can represent whole numbers in the range from -2147483648 to +2147483647.
>
> A **string** is formed of up to 255 ASCII characters.
>
> A **boolean** value is either FALSE or TRUE.

Variables are addressed by name. A variable name may contain up to 24 characters. It should begin with an alphabetical character but may contain digits after the initial character. The only characters that should be used are the letters A through Z, the underscore and the digits 1 through 9. Variable names are not case sensitive; therefore, VAR1 and var1 are considered to be the same variable.

The value of a variable is assigned with the Set Variable * ... script command. In the example:

> Set Variable * teststring from Expression "TheLine(CursorRow-1)",

teststring is the name of the variable, and the value given to it is the value returned by the function TheLine. One argument, CursorRow-1, is passed to TheLine. CursorRow is another function. It is possible to build sophisticated expressions by combining MicroPhone II variables and functions.

Space for the variable is created when it is given a value by the Set Variable * ... script command, as shown above. The variable retains its value until a new value is assigned to it. The variable remains in memory until you delete the variable or quit the program.

Abort Button

This button is in the horizontal bar at the bottom of the terminal window. It is visible only when a script is running, when sending text files or when pasting text into the terminal window. When sending text files, pasting text, or running a script, the button will read **Abort Send**, **Abort Paste**, or **Abort Script** respectively. Clicking the abort button terminates the operation in progress.

Alert Box

A window that appears on the screen to warn ther user of a problem or event that needs attention. Mouse and keyboard actions are not recognized until the user responds to the alert.

Batch

A method of processing in which similar items, such as files, are grouped and processed together. Batch files transfers allow users to transfer several files with a single command.

Baud

A measure of the number of changes of state per second on a communications channel. Several bits of information may be transmitted with each change in the channel's state. A 2,400 bps modem is actually a 600 baud modem that modulates 4 bits into each change in the channel's state, resulting in transmitting 2,400 bits per second. The use of baud in place of bps, albeit incorrect, has become so common that the terms are popularly considered interchangeable.

BBS

Bulletin Board System. A computer system that permits users to call in, read messages and exchange files and information.

Binary

A number system limited to 1s and 0s as opposed to the decimal system that uses ten digits (0-9). At the lowest level, all digital computers handle information in binary form..

Break

A signal sent to a remote system that is often used to interrupt communications or to cause a special action to be performed.

Button Bar

A MicroPhone II on-screen control panel for invoking scripts. The bar appears on the screen when there is at least one button to display.

Capture File

A file used to capture an incoming stream of text to the disk for later processing.

Cascading

In automatically establishing an XMODEM, YMODEM or YMODEM-G transfer, "cascading" refers to dropping down to less sophisticated protocols until a match is found with the remote computer.

Clipboard

A temporary storage area in memory used for moving graphic and textual information within the same document, between two different documents or between two different applications.