Spectrum Version 2.0 through 2.5.3 Addendum

This document describes what's new in Spectrum versions 2.0 through 2.5.3. If you are upgrading from Spectrum v2.1 or later, refer to Spectrum's online !Help! topic "Changes in Spectrum v2.5.3" and "Changes in Scripting v2.5.3" to learn about all the bug fixes and program feature changes made in the v2.2 through v2.5.3 updates.

Significant changes to Spectrum's scripting language were made in the v1.0 to v2.0 update, and additions were made in the Spectrum v2.1, v2.2, v2 and v2.4 updates. Bug fixes were made in the 2.5.2 and 2.5.3 updates. Rather than listing those changes on disk and having three separate script references, the changes have been repeated in this addendum. This also gives us an opportunity to correct an oversight in the v2.0 addendum…this addendum includes an index!

Spectrum is no longer distributed by My eSource, formerly Seven Hills Software. All sales are now handled by Syndicomm™. The references to Seven Hills in the 'Getting Started' and 'Scripting' manuals have been retained, as these reflect the status of Spectrum when those two manuals were first printed.

<http://www.a2central.com>
visit our web site for the gateway to your Apple][world!

<speccie@btopenworld.com>
*please direct all technical support questions
to this email address*

<http://www.ewannop.btinternet.co.uk/>
*visit the Spectrum web site keep in touch with
our latest updates!*

Table Of Contents

Script Language Changes

Notes

Installation Instructions

If you are a new owner of Spectrum: Follow all the installation instructions in the *Getting Started and Reference* manual. If you install Spectrum sounds you should also run the “SoundPatch” utility to add some new sound events into the Sound CDEV.

If you purchased the floppy disk version of Spectrum v2.2, you will need to order the Spectrum v2.5.3 Updater from Syndicomm™:

<http://www.a2central.com>

To learn about most of Spectrum, read the *Getting Started and Reference* manual. Once you are familiar with the original Spectrum features you can then read the “New or Changed Program Features” section below to find out what’s new in version 2.0 or later. Likewise, when you begin writing scripts you should read the *Scripting* manual to learn about the original Spectrum features, then read the “Script Language Changes” section below to learn what’s new in version 2.0 onwards.

If you purchased Spectrum version 2.2 on floppy disk:

✻ Insert the original Spectrum Install disk.

① Run the Installer, and when you are asked to do so, type your full name then click the OK button.
When personalization is complete, Apple's Installer is launched automatically.

② After installation, locate the Spectrum 2.5.3 updater archive, and run the Install script to update from Spectrum v2.2 to Spectrum v2.5.3.

If you purchased Spectrum version 2.5.3 on floppy disk:

✻ Insert the original Spectrum Install disk.

① Run the Installer, and when you are asked to do so, type your full name then click the OK button.
When personalization is complete, Apple's Installer is launched automatically.

If you purchased the CD version of Spectrum version 2.5.3:

✻ Drag the Spectrum folder to your hard drive to make the 'Spectrum.2.5.3' folder.

① Run the Installer, and when you are asked to do so, type your full name then click the OK button.
When personalization is complete, Apple's Installer is launched automatically.

If you will use Spectrum on a hard disk drive:

Skip to the next page if you use Spectrum with 3.5" disks.

③ Select the “Spectrum 2.5.3” script and click the Install button. Outdated files will be removed, and Spectrum v2.5.3 will be copied to the disk that was used to start up the computer. *NOTE: If you want to install Spectrum to some other disk, click the Disk button until the disk is displayed.*

④ After the update is installed, click the Quit button.

⑤ Launch the “SoundPatch” application and follow the on-screen prompts to update your Sound Control Panel (CDEV) so it lists *all* the currently defined “sound events.”

⑥ Choose Control Panels from the (Apple) menu, then open the “Sound” CDEV.

⑦ Assign the sounds you want by selecting an event from the Event popup menu, then choosing the sound to be associated with that event from the Sound popup menu. For example:

Event	Suggested Sound
File Transfer Complete	Send/Receive-Good **or** Ahh **or** Trumpets
File Transfer Failed/Abort	Send/Receive-Bad
Grabbing Screen	Saving Screen
Key Click	Key-Any
Modem Connected	Phone-Connected
Modem Hanging Up	Phone-Hangup
Modem No Carrier	Phone-No Connection
RealTime Message	Phone-Connected **or** Trumpets
Return Key	Key-Return

Space Key	Key-Spacebar
You Have Mail	You Have Mail **or** You Have Mail (HAL)

⑧ If you experimented in step ⑦ by choosing several sounds to hear what they sound like, choose Shut Down from the Special menu, then select the Restart option and click OK to restart the computer. The reason for restarting is that the Sound CDEV is now probably using a *lot* more memory than it needs to be, and restarting will fix that.

⑨ Now launch Spectrum version 2.5.3!

To install the !Help! NDA System and documentation for Spectrum:

① Select the ‘Spectrum Help System’ script and click the Install button. The !Help! NDA and related documentation will be copied to the ‘System:Desk.Accs’ folder on your boot drive.

② Reboot your hard drive to install the !Help! NDA into the Apple Menu.

If you use Spectrum with 3.5" disks:

See the previous section if you will use Spectrum with a hard disk drive.

③ Choose Quit from the File menu and the Finder reappears.

④ Make a backup copy of the Spectrum update disk.

⑤ Drag the original Spectrum update disk icon into the Trash to eject the disk and remove its icon from the desktop.

⑥ Insert the backup copy of the Spectrum update disk.

⑦ Rename the backup disk to be “Spectrum”.

⑧ Drag the following items into the Trash:

❑ Prefs.Convert
❑ Install
❑ Misc.

⑨ Choose Shut Down from the Special menu then click OK to shut down (turn off system power). You can now start using Spectrum v2.5.3 just as you have been using Spectrum v1.0 or later.

Feature Changes

This section describes the program feature changes made in all versions of Spectrum from Spectrum v2.0 through v2.5.3. Refer to Spectrum's online !Help! topic "Changes in Spectrum 2.5.3" to learn about the recent bug fixes and program feature changes made in the v2.2, v2.3, v2.4, v2.5.2 and 2.5.3 updates.

You will immediately see some changes with the Spectrum™ v2.5.3 menus. These are mainly to include the integrated support of Marinetti™ 2.0 and its TCP/IP environment.. This has led to a number of additional script commands to support TCP/IP through scripting. There are also a few other new menu items that have been added since Spectrum™ 2.1.

Underneath the surface there have been many changes, bug fixes, and new script commands added. Most of the additional script commands have been designed to enhance and improve script functionality, and many are a result of limitations found while developing both "Spectrum Internet Suite™" and other script sets. Some of the XCMDs have been expanded since the release of Spectrum v2.1. Some of these XCMD updates have already been issued with the various script sets, but the latest versions of all the current XCMDs are included with this release of Spectrum, and should be used in place of existing versions. Some existing XCMDs that are no longer required will be removed by the Installer.

The Spectrum™ 2.5.3 update did not introduce any major feature changes from Spectrum™ 2.2. A small number of script commands have been added however, and a number of minor bugs have been fixed.

General Changes & Bug Fixes

NOTE: The version 2.2 to 2.5.3 Preference file format has been extended to support new features. Version 2.x preference files are converted automatically to the new format. The new format cannot be used with older Spectrum Pref files.

- In v2.2, the code was significantly reorganized to make room for more features. The protocol transfer code and some of the dialog code have been put into two dynamic segments to be more memory friendly.
- The Spectrum Port driver wasn't forcing unused high-bits off (i.e., when a 7-bit data format was selected it was possible for the 8th bit to be set). Fixed.
- The Spectrum Port driver stopped liking DataLink modems, especially on ROM01 machines. Fixed.
- Made more refinements to try to reduce data loss:

  Removed a `_TickCount` call from the timer routine.

  Now waits for VBL to cycle when changing hardware handshake status.

  Added support for "II Not Disturb" to automatically turn "Master Time Keeping" on and off whenever Spectrum is started and quit.
- ^G (SysBeep) has been reworked so the correct Sound CDev sound should be played. Now the interrupt-friendly (raspy) beep will play only if Hardware Handshake is not checked and a carrier is detected.
- If an XCMD or Online Display is not loaded successfully, now a beep is sounded and the item's name is displayed for one second in the "Loading" status area.
- We added the capability for XCMDs to add custom menu items into Spectrum's menus. Several supplied XCMDs take advantage of this feature, including HodgePodge and Kermit.
- A new "" (solid black apple) menu has been added to the right of the Settings menu. The new menu is shown only when an XCMD has added an item to it.

- Fixed a problem where the menu bar clock would not change to “am” at midnight or “pm” at midday unless the clock was fully updated (e.g., by a window closing or other similar event).
- Alerts now use “the latest cool button layout.”

- From Spectrum™ v2.2 onwards:

  Spectrum™ now requires System 6.0.1. A hard disk is recommended.

  Improved script execution speed through a reorganization of the matching algorithm.

  Scripts can be compiled to optimise their execution speed.

  Spectrum™ and the XCMDs are now Y2K compliant, where appropriate, additional script commands have been added..

  Spectrum™ now features integrated support for Marinetti™ 2.0 TCP/IP.

  HTML files can be viewed directly from within Spectrum™.

  Internal support for the Babelfish™ Import and Export. Engine.

  New internal “Preferences” folder.

  Spectrum™ now looks first for the “Spectrum2.Prefs” file in the internal Preferences folder, before it looks in the System folder. This allows multiple copies of Spectrum to run smoothly.

  Conversion of old preference files no longer causes problems.

  Spectrum™ now supports the Control Panel version of Hierarchic. If not already active, Hierarchic will auto-load if the CDev is present in the CDevs folder.

  The Spectrum™ Port driver has been updated.

  Support for ‘finderSaysBeforeOpen’ calls to open Text files automatically.

- From Spectrum™ v2.3 onwards:

  An X/Y/Zmodem bug that slowed transfers and caused increased checksum errors has been fixed.

  A workround has been made for a bug in the Marinetti 2.0 TCP/IP stack. This could cause Spectrum to crash if a socket closed when more than 4K of data was returned.

  The RESUMENEXT script command now works correctly.

  Fixed a problem in the “Search File” script command that caused an endless loop under certain circumstances.

  Fixed a problem with sending Text files showing the wrong size in the Status dialog.

  Fixed some interaction problems with Hierarchic and custom menus.

  Fixed a Hangup problem when DCD and Hardware Handshake were not checked.

  Fixed the reversed paths for “Set SendPath” and “Set ReceivePath”.

  Improved the renaming algorithm when receiving duplicate files.

  Fixed a problem that would cause endless looping if the receive filepath was not found.

  Fixed a problem that could cause hanging if the Capture Buffer filled during batch downloading.

  Fixed a problem that could cause overwriting on the Text screen during downloads.

  Improved the handling of the WindowMgr Menubar switching.

  Fixed a bug in Evaluate where a number to the power of ‘0’ was returning an incorrect value.

  OA-G no longer clears the Text or VT100 screen display if they are already open.

  Improved recovery from cancelled X/Ymodem transfers.

  Fixed an intermittent lock up on printing.

- AppleWorks 5.0 documents include Inverse and MouseText characters. These will now display correctly within the Editor, though the Inverse attributes will be stripped. These characters will also be sent correctly if you send an AppleWorks file as text.
- There is a bug in the original GSPlus™ XCMD that can cause it to interfere with the messages being sent to other XCMDs. If you have been using this XCMD, you will find a new version supplied with Spectrum™ 2.5.3 that cures these problems.
- Version 2.5.3 fixed some problems with the TCP/IP data interface.

(Apple) Menu

- Removed Spectrum’s built-in help in favor of the !Help! NDA (which is included with Spectrum). If you don’t want any help you can inactivate the Help.NDA desk accessory (located in the System:Desk.Accs: folder).

  If you want help only while Spectrum is being used (even if you have Shift-booted the computer) you can move the Help.NDA into the same folder as the Spectrum application. If Spectrum loads the NDA, it is unloaded when Spectrum quits.

  Regardless of where the Help.NDA is located, the NDA always looks for its files in the Desk.Accs:Help.Files: folder.

- Added “Menu Item Help” that dims every menu item except those owned by Spectrum, and changes the mouse pointer to a “menu item help” pointer. When a menu item is selected, Spectrum asks the !Help! NDA to display the appropriate help information.

  NOTE: Spectrum cannot dim the TransProg III menu (if present). If you select an item from the TPIII menu, the help request is transmitted to !Help! but then TPIII will attempt to launch the item you selected.

File Menu

Open

- Option-opening a file of type $50 could cause a crash because Spectrum was interpreting all filetype $50 files as Teach files. Fixed.
- Loading an empty file into the Editor changed the name of the Editor window to the empty file’s name, even though the Editor contents were untouched after an error message was shown. Fixed.
- Holding the Option key down now allows the data or the resource fork (if present) of a file to be opened.
- If a file with the suffix ‘.HTM’ or ‘.HTML’ is opened, and the optional ‘HTML.Engine’ is present in the ‘Add.Ons:Drivers’ folder, you will be asked if you want to view the file as HTML. You will still see the file in its raw text form in the Editor after you have finished viewing the HTML display.

Save

- The correct filetype is now set on “saved as” files.
- With an autosave file set up, Save As will allow saving of the current buffer independently from the Save command.
- Added a clarification message when autosaving a buffer to a full disk.
- If you opened an AppleWorks file into the Editor, then File/Saved it, the file is saved as Text instead of AppleWorks. Fixed. (Use Save As to save as a different file type.)

Babelfish™

- Babelfish™ Import and Export is now supported. This requires Babelfish™ to be installed, and Text filters to be available. See the Babelfish™ documentation for further details.

Send/Receive

- Added the ability to transfer files that have a resource fork, by using a MacBinary header. Because the receiver must be capable of receiving and processing a MacBinary file, the default file transfer preferences are set so files with a resource fork cannot be transmitted. If you decide to change this preference, be sure the recipient is able to handle the file.
- If you received a file via Zmodem to a P8 volume, and the file had a period as the first character, Spectrum was doing strange things and reporting non-existent errors. This was caused by a bad JudgeName call. We also found that if you received a file by Zmodem that required JudgeName, the transfer did not complete properly. Both items have been fixed.
- A “Make this folder default” checkbox has been added to the file selection dialog boxes for all built-in file transfer protocols. When checked (and you accept a file to transfer) the folder that contains the file is stored as the default send or receive file transfer path. The choice is not written to the preferences file unless the settings are saved.
- When sending a text file, Macintosh text files would not be selectable. Fixed.
- When sending Xmodem ProDOS, the received file would be truncated. Fixed.
- Fixed a Zmodem escape character problem that showed on some host systems.
- The Freezer XCMD is no longer necessary, as its functionality has been incorporated directly into Spectrum.
- Spectrum was forcing filenames to lowercase for Zmodem and Ymodem receives. Fixed.

- If an incoming Zmodem file happened to have a colon or slash in its embedded filename, a “file not found” error would occur and the transfer would abort. Most Zmodem implementations do not pass these characters, but some do, so Spectrum now replaces those path separators with periods.

- Spectrum does not attempt to extract files from a Binary II wrapper if there are multiple files enclosed because it would be nearly impossible to resume a transfer. To help alert the user that he needs to use some other program (like ShrinkIt) to extract the multiple files, a message is now added to the capture buffer that says “WARNING: A Binary II header holding multiple files was not unpacked.”
- Removed the code to resume CIS B+ uploads because CompuServe has never implemented that capability.
- The file transfer status dialog box now shows whether a Binary II or MacBinary header is present on the incoming/outgoing file.
- When using CIS B+, if you typed only a filename to send or receive, the correct file transfer folder was not always set properly. Fixed.

Launch

- Fixed a problem that would not let the Hangup checkbox be unchecked.

Quit

- If you opened the SHR screen, then the Editor, then opened the VT100 display, then go back to the Editor and File/Quit, lots of windows would appear, disappear, reappear, etc. as the windows were closed in a fixed order. The windows are now closed in the order in which they are layered.

Edit Menu

Paste

- If text had been copied into the Clipboard, Paste was not highlighted when the Editor was first opened. Fixed.
- V did not work in VT100 if the chat line had been open in the Spectrum Text screen. Fixed.

Paste As Reply

- This feature didn’t work right if the selected text was greater than 76 characters. Fixed.

Find/Replace

- Pasting into the Find box now activates the Find button.
- The “apply/replace” script command was changing some choices in the dialog box. Fixed.

Signature

- If a Signature has been defined by the ‘Set Signature’ script command, this will insert that Signature into any open TextEdit window.

Show Menu

Online Display

- j did not work from some online displays; you had to press -Shift-J. Fixed so it is no longer case sensitive.
- When displaying echoed information, characters were routed to the script screen vector instead of the normal screen vector, which caused some information to be lost. Fixed.

Editor

- Added more support for extended keyboards (the small “forward delete” key, Home, End, Page Up, Page Down).
- If an NDA or other concurrent application sends a ‘finderSaysBeforeOpen’ call, and Spectrum™ gets the call first, the filetype will be checked to see if it is a Text, Teach or classic AppleWorks file. If so, it will be opened into the Editor. An example is double-clicking a Text file displayed in a Disk Access II™ catalog, when Spectrum will open the file before Disk Access II™ opens its default window.

Clipboard

- The clipboard window can now be printed. The window is also now “editable” in that you can highlight some text in the Clipboard window and Edit/Copy it…which deletes all of the clipboard contents and replaces it with the text you had highlighted.

Chatline

- If you press the down arrow in the first or second line when the cursor is to the right of the end of the next line, the cursor jumps to the end of the next line instead of sticking, and not moving as expected.
- If the chatline was full, pressing the Arrow keys would cause a beep. Now it beeps only when attempting to add a character..
- If you are using the SHR display, you can now move around the Chat Line by clicking with the mouse.

Phone Menu – TCP/IP Menu

Please refer to the documentation that comes with Marinetti™ for full details on how to setup and activate TCP/IP through the TCP/IP Control Panel.

Dial Number - Services

- Spectrum no longer auto-dials the next phone number if the current phone number fails to connect.
- Linking a script to a dialing entry no longer corrupts the first character of the filename.
- From Spectrum™ v2.3, Dial Number still operates as before, but if the optional TCP/IP Menu is selected instead, this item will be renamed to Services.
- The Modem Init dialog has been changed to show a “popup” menu of modem types instead of the original “Normal” and “High Speed” strings. The file that is used to build this menu is held in the new “Preferences” folder and so can be edited if you wish.
- “Services” leads to a similar dialog to the “Dial Number” dialog. You can create and edit a service list of your most used services. You can also connect directly to a system by entering an “IP name” or “dotted address” into the Connect box.
- “Telnet” mode can be optionally selected. In Telnet mode, the setting of the Half Duplex flag is ignored, and echoing to screen will be controlled by the host you are currently connected to.

Hangup - Connect - Disconnect

- In Serial mode, the ‘Hangup’ menu item is dimmed if offline.
- In TCP/IP mode, the ‘Hangup’ menu item is renamed to either ‘Connect’ or ‘Disconnect’. ‘Connect’ will make a connection to your ISP. ‘Disconnect’ will warn if you have any active connections, and then Disconnect you from your ISP.

Switch to TCP/IP - Switch to Serial

- Controls the switching between the two operating environments. When in Serial mode, the menu items are named as before, but when in TCP/IP mode, some menu items now have new names and functions:

Answer Back - Logout

- Answer Back operates as before.
- Logout logs you out from the currently active connection.

Service Items

- When you have logged in to a service, the connection or socket names will be entered at the foot of the TCP/IP menu. Each menu item is numbered, and you can then switch between active connections by selecting from this menu. You can also link these items to FKeys, by running the ‘TCPIP.FKeys’ script.

Script Menu

See the “Script Language Changes” section for the list of changes that affect scripts.

Learn a Script

- This will now record any pause from the last character being received, and a string being transmitted. This should help produce improved scripts where time sensitive actions are important. You should always subsequently edit generated scripts so they are optimised for best results.

Menu Files

- These files can now have optional pathnames that will be used instead of a linked “Letter” key. A space can also now be used instead of a “Letter” key. To build the file, ‘tabs’ are used to space any pathname that is given from the menu name. This now means that a Menu file might look like this:

```
A	First Script
B	Second Script with path<tab>:hard.disk:folder:filename
 	Third Script with path<tab>:hard2.disk:folder:filename
```

Settings Menu

Online Display

- The “Spectrum SHR Fast” and “Spectrum SHR Normal” displays have been combined into a single “Spectrum SHR” display that automatically switches between the fast and normal display modes as necessary.
- If Delete=Backspace is checked, character $08 is sent out the port when the Delete key is pressed, rather than whatever key sequence is associated with the left arrow key.

ANSI

- There was a slight possibility that commands with multiple parameters (e.g., ESC[31;47m) could be trashed. This has been fixed.
- The underline character attribute is now supported.
- Supports the direct actions ***Debug On*** and ***Debug Off*** to control whether unknown/unsupported Escape codes are displayed in the Status Line. For example:

```
DirectAction "Debug ON" 0
DirectDisplay "^[[8a"
```

- Escape sequences which do not have commands are ignored (or reported as invalid if display debugging is on).
- Invalid SM and RM commands are tolerated better.
- Supports Insert and Replace Mode (IRM) using the SM and RM commands with a parameter of 4 (e.g. ESC[4h) sets the mode so characters are *inserted* instead of *replaced*.
- Supports Horizontal Editing Mode (HEM) using the SM and RM commands with a parameter of 10 (e.g. ESC[10h) sets the mode so editing commands (insert/delete) go left instead of right.
- Supports the Insert CHaracter (ICH) command to insert a specified number of blank characters at the current location (respecting the HEM mode). For example, ESC[3@ inserts 3 blank spaces.
- Includes custom support for Genie’s Internet services.

Options

- The old “File Transfer Settings” submenu has been renamed. This better reflects its use by many of the XCMDs. For details of any additional Menu items, refer to the ‘Spectrum Extras’ subject, and the relevant sub-topic of the !Help! NDA.

The Options menu item now presents a sub-menu with Receiving Options and Sending Options and other Options items inserted by XCMDs. Most of the options existed in Spectrum v2.0; the new or changed items are as follows:

- SendAhead is relevant for both sending and receiving, but different protocols handle SendAhead differently. Because there is really only a single SendAhead option, if you change the setting in the Sending Options dialog box, it will also change in the Receiving Options dialog box (and vice-versa).
- The Sending Options dialog lets you specify the location of the files to be sent. The Receiving Options dialog lets you specify the default location for storing received files. In each case you can choose no default location, and files will be sent from/received into the most-recently used folder.

- The Receive Options dialog lets you specify the filetype and auxtype to be assigned when the received filetype/auxtype is unknown.
- MacBinary is now supported so files with a resource fork can be transmitted to other Spectrum v2.4 owners, to Macintosh computers, or to others who can process a MacBinary header.

.

External Commands (XCMDs)

The “Spectrum Extras” subject and sub-topics in the !Help! NDA contain complete documentation on the new and updated XCMDs provided with Spectrum v2.4.

NOTE: The Freezer XCMD is no longer necessary because its automatic features are included directly in Spectrum v2.1. It can still be used if your scripts want to freeze the mouse manually.

The following XCMD-related changes were made to Spectrum itself:

- Unclaimed control hits might not have been seen by XCMDs correctly. Fixed.
- The version number in Spectrum’s `_AcceptMessage` call has been changed to 2.1.
- The BinHQX XCMD has been expanded since Spectrum™ 2.1, to include Base 64 and UUencode functions.
- NOTE: The TopCat, BatchXfer and Freezer XCMDs are no longer necessary because their features have been incorporated directly into Spectrum™ v2.4.
- The ResEdit XCMD has been replaced by the WorkBench XCMD .

TCP/IP

Spectrum now has two modes of working. The original Serial connection mode and a new TCP/IP mode. You must have the Marinetti™ TCP/IP Cdev installed and configured to use the TCP/IP mode.

If you regularly use any other communications programs, you may wish to use Serial mode as your default setting when Spectrum first starts up. If TCP/IP has made an active connection by another application before Spectrum is started, TCP/IP mode will be forced, regardless of your default settings. If you wish to continue with a serial connection, you must first Disconnect TCP/IP by either selecting Switch to Serial on the TCP/IP menu, or by calling the *Set Serial* script command.

If the TCP/IP CDev or another application makes the connection, Spectrum will be forced into TCP/IP mode when it starts. You will see an Alert box warning you of this.

When you Quit Spectrum, if TCP/IP is still *Connected*, any open sockets will be closed automatically whether TCP/IP is disconnected or not.

TCP/IP Menu

Services

Choose *Services* to *Connect* to a host or to create Service entries for the systems you call.

Services Entries

Click *New* to create a new Service entry; to edit an existing entry highlight it and click *Edit*.

Up to 32 Service entries can be created within the list. Each Service entry remembers the dotted address or system name that you enter, an optional linked script, and the required display. If you choose *'Default Display'*, then the current display will be used. Unlike the phonebook entries, a Service entry does not remember other settings, as these are not directly relevant to a TCP/IP connection. The Service list is stored as a text file in the '@:Spectrum:Add.Ons:Preferences:' folder. This file can be manually edited if you wish. The 'Tab' separated fields for each line are: *Entry Name, Address, Script and Online Display*.

Login

Double click an entry from the list, or enter the dotted address or host name into the Connect box. If you are not already online you will automatically be connected.

Telnet

The Telnet mode is the default mode used for most services. It will be automatically set whenever you open the Services dialog. If you do not wish to connect in Telnet mode, then uncheck the 'Telnet' box before logging in. Note that the setting of *TCPEOLTranslation*, *TCPFlushChar* and *TCPFlushFrequency* will all affect the keyboard responses if Telnet mode is turned off.

TIP: If the connection fails, and you cannot Disconnect normally, you may find that the modem is still online. Quickly switch to Serial mode, select the same baud rate that TCP/IP was using, and select Hangup to drop the line.

TIP: If you do not want to see messages while you connect, or do not want the Capture buffer and screen to record a switch of services. uncheck the respective checkboxes.

Connect - Disconnect

If you are not yet connected to your ISP, the menu will show *Connect*, and will connect you to your ISP. Choose *Disconnect* to logout ALL active Connections and Disconnect from the host. If there are Connections active, you will be warned and have the option to Cancel.

Once connected, you will still need to login in to a service from the Services dialog.

Send Break

This menu item will be dimmed while TCP/IP is active.

Switch to Serial

This menu item will switch to the Serial mode from TCP/IP mode.

Close Connection

Choose *Logout* to logout from the currently selected *Connection* shown by the √ marker. If there are other active Connections, then the first of these will then be selected as the new active Connection.

Connections:

Each connection you open will add an entry to the end of the TCP/IP menu. If the chosen Connection was opened from an item in the Service List, or the optional MenuName was used in the *Open TCPSocket* script command, then that name will be used for the Menu item. Otherwise the domain name or dotted address will be used. Each entry is numbered to show the logical Connection.

You can change active connection by selecting a new one from those shown. A √ will show against the current active *Connection*.

When you switch *Connections*, the current display type, screen contents, and up to 32K of the *Scrollback* Buffer will be preserved. If you then switch back to this *Connection* later, the display will be restored as it was when you last viewed it. If a script has set the *'Set Screenbypass ON'* flag, then the displays will not be reopened automatically. Note that with a large number of sockets open, you could eventually run out of memory if all the *Scrollback* Buffers were full. It is advised to size the *Scrollback* Buffer to suit the number of *Connections* you might expect to use.

When you switch *Connections*, a marker line will be added to the *Capture* buffer, to show a new *Connection* has been selected.

Tip: Define your FKey Menu items for the first ten connections. You then be able to quickly switch these by using the OA-# equivalent.

Script Language Changes

This section details new and changed script commands in all versions of Spectrum between 2.0 and 2.4. It is intended for script *authors;* script *users* should find that scripts written for version 1 still execute identically under version 2.4 (just a whole lot faster).

NOTE: Items marked with a bar to the left of the text (like this paragraph) indicate changes made from v2.0 to v2.4. If you are already familiar with v2.0, focus on the marked items.

General

- Instead of having only ten variable numbers (0 through 9), Spectrum now supports any number of *named* variables.
- Added support for “Spectrum External Commands” (or XCMDs), which can provide sophisticated add-on features for scripts.
- Version 2.4 executes scripts many times faster than version 1 did!
- The chosen online display automatically opens for commands that require it to be open, but the flag wasn’t getting reset if the Editor window was opened. Fixed.
- “#” comments at the end of commands with optional parameters could cause problems. Fixed.
- If a line had only a replacement variable on it, and the replacement variable is empty, an error would occur. Fixed.

Specially-Treated Characters

$ (Replacement Items)

$Version

Gets replaced by the software’s name and version number (e.g. Spectrum 2.1).

$AutoSaveFile

Gets replaced by the name of the current AutoSave file (if the filename has had a suffix added or incremented, this replacement will show the current name).

$Debug

Gets replaced by the current debugging state (see Set Debug in the scripting manual).

$DebugFile

If *$Debug* indicates debugging is going to a file, then *$DebugFile* provides the *FoldernameFilename* of the debug file.

$LastFile

Gets the name of the last file that was saved or selected in a standard file dialog.

$LastXferPath

Gets replaced by the *Foldername* of the last successful file transfer. The value remains valid until the next successful file transfer. *NOTE: The Kermit XCMD does not update this value.*

$LastXferFile

Gets replaced by the *Filename* of the last successful file transfer. The value remains valid until the next successful file transfer. *NOTE: The Kermit XCMD does not update this value.*

$Date
$DateTimeStamp

- If the clock is set to the year 2000 or beyond, $Date and $DateTimeStamp would fail with a random error message. Fixed.

$LongYear

Gets replaced by the four digit year (e.g., 1996) so scripts can be written to handle the year 2000 and beyond.

$ReceiveFType

Gets the filetype (in decimal) that is assigned when a received file's filetype/auxtype is unknown.

$ReceiveAType

Gets the auxtype (in decimal) that is assigned when a received file's filetype/auxtype is unknown.

$Trigger

Returns the value of the last trigger string that was seen. Returns 0 if no triggers have been seen since the start of the script. Typically this replacement item would be used only in a trigger handler routine.

$Buffer

Gets replaced by the size of the contents of the capture buffer.

$FullTime

This now strips any trailing spaces (applies only to 24-hour time format).

$ErrorMsg

Some of the messages have changed for clarity. For example, “Parameter not assigned” is now “Required parameter is missing.”

$ErrorCode

Only valid after a disk access error occurs

Gets replaced by the hexadecimal code for the disk access error that occurred. See the “Error.Codes” file in the Documentation folder for a list of possible errors.

$FilePos#

# must be replaced by a number from 0 to 9

Gets replaced by the character position of the “expected” file marker for the specified file number. If $FilePos3 is 7 then the next character that is read from file 3 will be the 7th character in the file; if a character is written then the 7th character in the file will be overwritten.

$ActualFilePos#

# must be replaced by a number from 0 to 9

Gets replaced by the actual file marker position for the specified file number. This value will match the $FilePos# value unless a Set FilePos command has been used (in which case $ActualFilePos# shows the *true* file position, while $FilePos# shows the *desired* file position).

$EOF#

# must be replaced by a number from 0 to 9

Gets replaced by the character position of the last character in the specified file (the End Of File marker).

$EditorSize#

# must be replaced by a number from 0 to 9

Gets replaced by the number of characters in the given script editor.

$EditorFile

Gets replaced by the name of the file in the editor (if the editor has been saved).

$LastScriptPath

Gets replaced by the *Foldername* of the script which chained to the currently-running script. If the currently-running script was not chained to, or was chained to from a script in the Editor, then *$LastScriptPath* is "".

$LastScriptFile

Gets replaced by the *Filename* of the script which chained to the currently-running script. If the currently-running script was not chained to, or was chained to from a script in the Editor, then *$LastScriptFile* is "".

$ReplyQuote

Gets replaced by the quote string (see Set ReplyQuote).

$Quote

Gets replaced with the current quote character (see Set Quote).

$Token

Gets replaced with the current token character (see Set Token).

$Baud

Gets replaced by the current baud rate, in the format used by the Set Baud command (i.e. 2400 or 9600).

$DFormat
Gets replaced by the current data format, in the format used by the Set DFormat command (i.e. 7E2 or 8N1).

$Duplex
Gets replaced by the current duplex setting, in the format used by the Set Duplex command (i.e. HALF or FULL).

$Echo
Gets replaced by the current echo setting, in the format used by the Set Echo command (i.e. OFF or ON).

$StatLine
Gets replaced by OFF or ON to indicate the state of the status line.

$ChatLine
Gets replaced by OFF or ON to indicate the state of the chat line.

$SendLines
Gets replaced by the line break position, as set by the Set SendLines command.

$Hit
Gets replaced by the number of the last Hitzone that was clicked.

$Update
If a screen update occurred on a super hires display, $Update is set to 1. This is useful for advanced scripts that may need to redraw special screen elements (rectangles, icons, etc.).

$Dialed
Intended for use in a logon script. Gets replaced by a null string if no phone number was dialed (i.e. the phonebook entry did not have a phone number attached). It is replaced by “Yes” if a phone number was dialed.

$PBMethod

Gets replaced by a “T” or “P” to indicate Tone or Pulse dialing, as set by the last-accessed phonebook entry.

$PBNumber

Gets replaced by the phone number of the last-accessed phonebook entry.

$PBDisplay

Gets replaced by the name of the Online Display that is attached to the last-accessed phonebook entry.

$PBRedial

Gets replaced by the redialing options set for the last-accessed phonebook entry.

$PBFile

Gets replaced by the *Filename* of the script (if any) that is attached to the last-accessed phonebook entry. *($LogonFile* is still valid for compatibility with version 1.)

$PBPath

Gets replaced by the *Foldername* of the script (if any) that is attached to the last-accessed phonebook entry.

$SEDirty#

# must be replaced by a number from 0 to 9

Gets replaced by 1 if the specified script editor has been changed since being loaded or saved; null if the editor has not been changed.

$EditorDirty

Gets replaced by “1” if the editor contents need to be saved; otherwise it is null.

$BufferDirty
Gets replaced by “1” if the capture buffer needs to be saved; otherwise it is null.

$AutoSaveBuffer
Gets replaced by the state of the AutoSaveBuffer flag.

$EditorHandle#
# must be replaced by a number from 0 to 9

Gets replaced by the memory address of the TextEdit handle that holds the data for the specified script editor. For use when communicating with External commands.

$FailureCode
After a command that waits for port input, if Failed is true then $FailureCode gives the reason the command failed:

0 = Other source (e.g. a Hitzone)
1 = Timeout was reached
2 = IdleTimer was reached

$DoHangup
After using the Get File command with a Kind of 3 (launchable applications), $DoHangup indicates whether the “Hangup” option was checked (1 means yes; null means no).

$DoReturn
After using the Get File command with a Kind of 3 (launchable applications), $DoReturn indicates whether the “Return to Spectrum” option was checked (1 means yes; null means no).

$FreeMem
Provides the number of bytes of real free memory currently available.

$DisplayOpen

Gets replaced by ON or OFF depending on whether the current online display is open or closed.

$FileID#

# must be replaced by a number from 0 to 9

Gets replaced by the GS/OS ID of an open file. The returned value cannot be directly used within any script commands. It is intended for use by XCMDs that can support the direct handling of an open file.

$Date2

Gets replaced by the date using a LongYear value. (e.g. 3 Nov 1993, 17 Sep 2000)

$DateTimeStamp2

Gets replaced by the date/time string using the LongYear value. (e.g. D17Sep1994T1040)

$Year2

This is the same as $LongYear. And is provided for compatiblity with the new LongYear replacement values.

$ChainLevel

Gets replaced by the current level in the Chainback sequence. The returned value will be from 0-16.

$Signature

Gets replaced by the Signature string if it has been defined.

$BufferSize
$ScrollSize
$PortSize

Gets replaced by the size of the defined buffers. Note that this is the maximum size of the buffer as defined, not the size of its current contents.

$BufferSaved

Gets replaced by a NULL string if the buffer has been saved, or a ‘1’ if not. Use this in conjunction with the $BufferDirty variable.

$TCPInstalled

Gets replaced by ON or OFF depending on whether the Marinetti TCP/IP Cdev is installed or not.

$SerialActive

Gets replaced by ON or OFF depending on whether the serial mode has been selected or not

$TCPActive

Gets replaced by ON or OFF depending on whether the TCP/IP mode has been selected or not

$TCPOnline

Gets replaced by ON or OFF depending on whether TCP/IP is connected or not.

$TCPError

Gets replaced by the last error to be reported by Marinetti when a TCP/IP command failed.

$TCPIPID#

# must be replaced by a number from 1 to 32

Gets replaced by the socket number used by scripts, from a logical connection number used by manual connections. This allows a script to pick up and use any sockets that have been logged into manually.

$ActiveSocket

Gets replaced by the ID # for the current TCP/IP connection. This value can be used subsequently in some of the TCP/IP script commands.

$FreeSockets

Gets replaced by the number of free sockets available. The returned value will be in the range of 0 to 32.

$TCPConnectName

If TCP/IP is active, this returns the name of the current Link Layer set within the TCP/IP CDev.

$TCPMessages

Gets replaced by ON or OFF depending on whether messages are currently being displayed for TCP/IP actions.

$TCPMode

Gets replaced by a value representing the current mode set for TCP/IP.

1 = data returned through port vectors
2 = data returned through port vectors (Telnet mode)
3 = data only returned through Get TCPData and Get TCPHandle

$TCPNotify

Gets replaced by ON or OFF depending on whether switching between active sockets is recorded in the capture buffer.

$IPAddress

If TCP/IP is active and online, gets replaced by the local address of the TCP/IP stack in the format "123.456.789.012".

$EOLChar1

Gets replaced by the first character of the two EOL characters, range 0-255.

$EOLChar2

Gets replaced by the second character of the two EOL characters, range 0-255.

$FlushChar

Gets replaced by the character used to flush a line, range 0-255.

$FlushFrequency

Gets replaced by the value used to determine the maximum length of line that will be transmitted, range 1-512.

Parameters

VarNum becomes *Varname*

Spectrum now supports *named* variables—there is no longer any such thing as a “VarNum”! Every place you see “VarNum” in the script manual, substitute “Varname”.

Spectrum still supports the ten numbered variables (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9). With the exception of those default variables, variable names must start with a letter.

Variable names can contain any combination of letters and numbers, up to 32 characters long. Uppercase/lowercase does not matter (i.e., MYNUM, mynum, MyNum, and mYnUm are all the same variable, regardless of the CaseSensitive setting).

The only naming restriction is that you cannot create a variable based on the name of a built-in replacement item. For example, you cannot use “Date” because it is a built-in replacement item, nor can you use variable names that begin with Date (e.g., “DateOfCall” or “Date2”). You may, however, use names such as “MyDate”. *NOTE: An “Assigned name is reserved” script error occurs if you attempt to use an existing replacement name.*

Named variables work just like the numbered variables work: any place a *VarNum* was accepted, a *Varname* can now be used. The only minor difference between the two kinds of variables is that variables 0-9 are always defined and pre-set to null ("") by default, but named variables are not defined until they are created.

This means that when running a fresh script, displaying $0 will show nothing (because var 0 is pre-set to ""), but displaying $Something will show “$Something” on the screen because a variable named “Something” has not been created yet.

New named variables are created explicitly (e.g. Set Var MyNamedVar "Hi there!") or implicitly (e.g. Multiply 41 8 TheResult). Simply referring to a variable (i.e. using "$SomeVariable") does not create it.

In summary, *numbered* variables and *named* variables are nearly identical. The differences are:

- *Numbered* variables always exist; *Named* variables exist only after they are created.
- Certain commands (such as Store Vars, Restore Vars, Clear Vars, etc.) operate only on the ten *numbered* variables.
- Certain commands (such as Delete Vars) work only on *named* variables.

Tips for Advanced Users

When setting a variable (Set Var MyColor "Blue") or accessing its contents ($MyColor) Spectrum must first find the exact variable being referenced before it can work with it.

Initially the variable name table is very short, but as you create new variables the table of variable names is expanded. The more variables you have, the longer the table is, and the longer it takes to search the table.

The search speed is not a concern unless you define hundreds of variables, but even then the search can be optimized by using shorter variable names, or long names that are unique in the first few characters.

For example, using variables named “My1VariableName” through “My99VariableName” will be faster than using variables named “MyVariableName1” through “MyVariableName99”. And using “M1” through “M99” will be even faster.

If you anticipate defining many variables, consider combining several pieces of information into a single variable. For example, instead of having 100 different variables to remember a “yes/no” response you could use a single variable to remember each response (e.g. “YYNNNYYN…”) and use the Substring/Overlay String commands to get and set the appropriate character.

Also, to keep the variable name table short, use the Delete Variables command to delete any variables created for temporary use.

String

A *String* can now contain up to 256 *Characters* (the previous limit was 128). Accordingly, this means that Variables, FKeys, Passwords, etc. can store up to 256 characters each.

Script Development

- Extra spaces, commas, and hard spaces no longer affect command recognition (e.g. “if   contains” is now recognized as being the command “if contains”).

Set Debug *State {"FoldernameFilename"}*
State can be External, File, Screen, Scrollback, or Off
FoldernameFilename is required when *State* is File, and must not be present for all other *States*

This command works like it did in version 1, with these additions:

- When Spectrum is executing a script returned from an XCMD, the debug statement shows “XCMD –=>” instead of “DEBUG –=>”.
- “Set Debug External” broadcasts debugging information to Spectrum External Commands, in case there is an XCMD that wants to handle debugging in a custom way. A “Debug” XCMD is provided as an example to show what is possible.
- “Set Debug File” directs Scrollback data into the specified *FoldernameFilename* instead of to memory. This option significantly slows down script execution because the file is updated frequently. It is recommended the file be created on a RAM disk or a fast hard drive.
- If “Set Debug File” was active and the disk gets full, subsequent file saves would result in an invalid reference number error. Fixed. Now if the disk gets full, the file is closed and debugging is turned off.

Set DebugTimeInfo *State*

State can be Off or On

When debugging with this setting turned on, the debug statement includes time information which can be used to optimize scripts.

Fundamental Commands

Clear Screen

Clears and resets the screen as if File/New was chosen.

DirectTransmit "*String*"

Transmits *String* directly out the port, without giving the Online Display any opportunity to alter the string (unlike the *Transmit* command which sends outgoing characters through the Online Display).

Transmit "*String*"

Shortcut: **Xmit "*String*"**

- If Duplex is Half and Set Screen is Off, then this will not open the online display.
- Now complains if too many quotes appear.

Clear Clipboard

Clears the Clipboard of any content. An empty Clipboard can speed up the operation of some script commands.

Open Help *Value1 Value2 Value3*

Optional: *Value1* = Subject, *Value2* = Topic, *Value3* = TargetString

Opens the !Help! NDA, and if the optional Subject, Topic, TargetString are specified, will attempt to scroll to that position.

Close Help

Closes the !Help! NDA.

Set FKey Menu *Value MenuValue*

Value is a number from 0-9 for the OA-Fkey, *MenuValue* is a number from 1-51

Assigns specific menu items to an FKey instead of the usual “String”. This allows those menu items to be fired as easily as if they had a Key Equivalent. The file transfer items are probably the most useful items that can be assigned

MenuValues	***Phone-TCP/IP Menu***
1	**Switch TCP/IP**
20-51	**TCP/IP sockets 1-32**
MenuValues	***Receive File Menu***
2	**CIS B+**
3	**XModem**
4	**1K XModem**
5	**4K XModem**
6	**YModem Batch**
7	**YModem-g**
8	**ZModem**
MenuValues	***Send File Menu***
9	**CIS B+**
10	**XModem**
11	**1K XModem**
12	**4K XModem**
13	**YModem Batch**
14	**YModem-g**
15	**ZModem**
MenuValues	***BabelFish Menu***
16	**Import**
17	**Export**

Play Sound "*Name*"

This command now plays any sound that can be seen by the resource manager, not just sounds stored in the *:System:Sounds: folder. *NOTE: Resource names are case sensitive.*

Play Event *Value*

Plays the sound for the specified event at the volume specified in the Sounds Control Panel. *NOTE: In order to play sounds the Sounds CDEV must be installed and active, and the "Sounds" checkbox must be on in the Online Display Settings dialog box.*

Example:

```
Play Event $$8 # bad keypress
Play Event $$100 # You have mail
```

The standard Sound CDEV does not list all these sound events. Run the "SoundPatch" utility to add listings for most of these events.

Sound Events	
0000 Alert stage 0	0053 Alert note
0001 Alert stage 1	0054 Alert caution
0002 Alert stage 2	0060 Screen blanking
0003 Alert stage 3	0061 Screen unblanking
0004 Outside window	0070 Beginning long operation
0005 Operation complete	0080 Application launching
0008 Bad keypress	0081 Application quitting
0009 Bad input value	0100 You have mail
000A Input field full	0E00 Error window base
000B Operation impossible	0EFF Error window other
000C Operation failed	0F00 Visual sound
0011 GSOS to P8	0F80 File transferred
0012 P8 to GSOS	0F81 Real time message
0013 Disk inserted	0F82 File transfer failed
0014 Disk ejected	1000 Connected to service
0015 System shut down	1001 Disconnected from service
0016 Volume changed	1002 Entered real time chat
0030 Disk request	1003 Left real time chat
0031 System startup	1004 Entering Forum/RT area
0032 System restart	1005 Leaving Forum/RT area
0033 Bad disk	1008 Modem-Dialing
0034 Key click	1009 Modem-Hanging up
0035 Return key	100A Modem-No carrier
0036 Space key	100B Modem-Connected

0040 Whoosh open	1010 Feature enabled
0041 Whoosh closed	1011 Feature disabled
0042 Fill trash	1012 Taking screen shot
0043 Empty trash	1020 Reading message
0050 Alert window	1021 Sending message
0052 Alert stop	

Settings

Save Settings "*FoldernameFilename*"

FoldernameFilename is optional

Saves Spectrum’s current settings to disk If no file is specified, settings are saved to the current settings file. If a file is specified then settings are saved to that file, and it becomes the current settings file.

- Would not save to a file unless it already existed. Fixed.
- Neither filters nor passwords were being copied over to the new settings file, and a subsequent Load Settings would not restore the phonebook.

Load Settings "*FoldernameFilename*"

FoldernameFilename is optional

Loads Spectrum’s current settings from disk. If no file is specified, settings are loaded from the current settings file. If a file is specified then settings are loaded from that file, and it becomes the current settings file.

Set ScrollData *Kind*

Kind can be Raw or Filtered

Determines how incoming data is stored into the Scrollback buffer. Raw stores the actual data that was received; Filtered stores data after it has been processed by the online display (i.e. the same information that appears in the Capture Buffer).

Set ReplyQuote "*String*"

String can be 1-16 characters

Sets the quote format used when choosing Paste As Reply from the Edit menu.

Online Display Settings

Set OnlineDisplay "*Display*"

- The Failed flag was not being set or cleared correctly to indicate whether this command succeeded. Fixed.
- This command now clears any defined Hitzones.
- Setting *Display* to “Spectrum SHR Fast” or “Spectrum SHR Normal” actually selects the combined “Spectrum SHR” display.

Close OnlineDisplay

Shortcut: **Offline**

Closes the current online display and displays Spectrum’s 640 mode desktop.

IMPORTANT: This command no longer causes the script to stop (because scripts can now operate even when the Online Display is not open). For full compatibility with version 1, edit any version 1 scripts so that “Close OnlineDisplay” is followed by the “Stop Script” command.

Clear Desktop

Closes all Spectrum windows and NDAs, displaying Spectrum’s 640 mode desktop.

Set NullStrip ON/OFF

Controls the removal of nulls in the input stream to the screen. The default setting is ON.

File Transfer Settings

Set RelaxedXfers *State*

The state of this setting is now stored in the preferences file, if settings are saved.

Set FileXferPath "*FoldernameFilename***"**

Sets both the ReceivePath and SendPath locations to the specified folder. Setting to an empty path ("") clears the paths and files will be received into/sent from the most recently-accessed folder.

Set MacBinary *Status*

Status can be On, Off, NoRez, Rez, Downloads

Sets the MacBinary setting, and can alter the Binary2 settings. Refer to the following table:

Script Command	Binary2 Down	MacBin Down	Binary2 Up	MacBin Up	Allow Rez
Binary2 On	-	-	sets	clears	clears
Binary2 Off	clears	-	clears	clears	clears
MacBinary On	-	-	clears	sets	sets
MacBinary Off	-	clears	-	clears	clears
MacBinary Rez	-	-	-	-	sets
MacBinary NoRez	-	-	-	-	clears
Binary2 Up	-	-	sets	clears	clears
Binary2 Down	sets	-	-	-	-
MacBinary Down	-	sets	-	-	-

Set ReceivePath "*FoldernameFilename*"

Sets the default path used when receiving files. Setting to an empty path ("") clears the path and received files will be stored in the most recently-accessed folder.

Set SendPath "*FoldernameFilename*"

Sets the default path used when sending files. Setting to an empty path ("") clears the path and files will be sent from the most recently-accessed folder.

Set ReceiveFType *Value*

Value can be 0-255 ($$00-$$FF)

Sets the filetype to assign when a received file's filetype/auxtype is unknown.

Set ReceiveAType *Value*

Value can be 0-65535 ($$00-$$FFFF)

Sets the auxtype to assign when a received file's filetype/auxtype is unknown.

Set AutoReceive *State*

State can be Off, On, BPlus, or Zmodem

Controls the “Auto B+” and “Auto Z” checkboxes.

Off: Turns off both checkboxes.
On: Turns on both checkboxes.
BPlus: Turns on the BPlus checkbox.
Zmodem: Turns on the Zmodem checkbox.

Set BinaryII *State*

State can be Downloads, Uploads, Off, or On

Works as described in the Scripting manual, except Downloads and On will no longer uncheck the “Resume Transfers” checkbox (because resume now works regardless of the receiver’s BinaryII setting).

Set SendLines *State*

State can be Off or On, or a number from 10-65535

Controls how text files and ScriptEditors are sent. Off sends files as-is; On splits paragraphs into lines of approximately 73 characters each; a Value splits paragraphs into lines of approximately the specified number of characters.

Dialing

Dial Service "*PhonebookEntry*"

Instead of generating a script error, now the Failed flag is set if the specified phonebook entry does not exist.

Dial Entry *Value*

Instead of generating a script error, now the Failed flag is set if the specified phonebook entry does not exist.

Get ServiceInfo "*PhonebookEntry*"

Establishes the port settings for the specified phonebook entry, and updates all the *$PB_* replacement items so they reflect the information stored in the phonebook entry. The Failed flag is set if the specified phonebook entry does not exist.

Connected

Scripts should issue this command any time they establish a connection with another computer or modem.

Onhook

Scripts should issue this command any time they close a connection (or when they know a connection has been broken) with another computer or modem.

TCP/IP Commands

Note: When in Telnet mode, the setting of Half Duplex will be ignored.

Switch TCP/IP

Disconnects the Serial Port and switches control to the TCP/IP environment. Note that this may terminate an active Serial connection.

TCPConnect

If currently offline, makes a TCP/IP link through the the TCP/IP CDev. It simply makes the initial link to your ISP, it does not open a socket Connection to a target host. If TCP/IP was already linked, this command will do nothing and will not set the Failed flag. The Failed flag will be set if the link failed.

TCPDisconnect

If currently online, will disconnect all active Connections and then disconnect the line from your your ISP. If TCP/IP was already disconnected, this command will do nothing and will not set the Failed flag. The Failed flag will be set if the command failed to disconnect.

Open TCPSocket *"AddressString" Value VarName MenuName*

"AddressString" is a dotted address in the format "123.456.789.123:23" or
"AddressString" is an address in the format "delphi.com"
Value can be from 1 to 3 (Mode)
  1 = data returned through port vectors
  2 = data returned through port vectors (Telnet mode)
  3 = data only returned through Get TCPData and Get TCPHandle
Varname returns a Socket number
The optional *MenuName* is a String maximum 18 characters

Links if not already Linked (See *TCPConnect*), and then opens a connection to the IP address and port in "*AddressString*" using the Mode given in *Value*. Up to 32 connections may be open at once. See documentation for TCP/IP for further details.

If the optional *MenuName* is given, it will be used to name the inserted menu item for that connection. The Timeout setting will be respected, with a maximum wait of 60 seconds. The Failed flag will be set if the Connection could not be opened.

Close TCPSocket *Value*

Value must be a valid Connection Number (Socket)

Closes the Connection. Sets the Failed flag if the Connection was not open or could not be closed.

Check TCPSockets

Checks through all the active Sockets, and if any have been closed, and there is no data waiting to be sent or received, will log them out and return any *TCPClosedResponse* strings that have been set.

Set TCPActiveSocket *Value Value1*

Value must be a valid Connection Number (Socket)
The optional *Value1* can be from 1 to 3 (Mode)

Changes the active connection for the data streams. This allows you to control a number of connections simultaneously. Sets the Failed flag if the Connection is not open.

Get TCPData *"LineEndMarkerString" VarName Socket*

"LineEndMarkerString" is a unique set of characters
Control and Quote characters are stripped from the returned *Varname*.
The Socket is optional.

TCP/IP must be active and an online connection must be made before this command will complete. Returns up to 256 characters from the active *Connection* or the optional *Socket*. When the *LineEndMarkerString* is received, the line is stored into the specified *Varname*. The *LineEndMarkerString* is not included in the returned *Varname*. You may wish to set *Mode* 3 for use with this command. Sets the Failed flag if not online or no data was waiting.

Get TCPEditor "LineEndMarkerString" EditorNumber Socket

"*LineEndMarkerString*" is a unique set of characters
Full 8 bit data integrity is retained in the returned *EditorNumber*
The *Socket* is optional.

TCP/IP must be active and an online link must be made before this command will complete. Accepts characters from the active *Connection* or the optional *Socket*. When the *LineEndMarkerString* is received the line is stored into the specified *EditorNumber*. The *LineEndMarkerString* is not included in the returned *EditorNumber*. You may wish to set *Mode* 3 for use with this command. Sets the Failed flag if not online or no data was waiting.

Get TCPHandle EditorNumber Socket VarName

Full 8 bit data integrity is retained in the returned *EditorNumber*
The *Socket* is required.

TCP/IP must be active and an online link must be made before this command will complete. Accepts characters from the designated *Socket* or the active connection if Socket = 0. If no data was waiting, the *Editor* will be returned empty. The *VarName* returns the current *Status* value for the socket. You may wish to set *Mode* 3 for use with this command. Sets the Failed flag if not online.

Send TCPData *"String" Socket*

Socket is optional

Sends the *String* directly to the active *Connection* or the optional *Socket*. No Telnet translation is made by this command. Sets the Failed flag if not online.

Send TCPEditor *EditorNumber Socket*

Socket is optional

Sends the *EditorNumber* directly to the active *Connection* or the optional *Socket*. No Telnet translation is made by this command. Sets the Failed flag if not online.

Flush TCPSendBuffer

Flushes any waiting data in the *TCPSendBuffer* to the active *Connection*. Data is normally flushed at each character in Telnet mode, or when the *TCPFlushChar* being seen in the output flow. This command allows any waiting data to be flushed before that character is seen. Sets the Failed flag if not online.

Get TCPStatus *Value VarName*

Value must be a valid Connection Number (Socket)
Returns the Status of the Connection:

0 = tcpsCLOSED	1 = tcpsLISTEN
2 = tcpsSYNSENT	3 = tcpsSYNRCVD
4 = tcpsESTABLISHED	5 = tcpsFINWAIT1
6 = tcpsFINWAIT2	7 = tcpsCLOSEWAIT
8 = tcpsLASTACK	9 = tcpsCLOSING
10 = tcpsTIMEWAIT	

See the TCP/IP documentation for more details. Sets the Failed flag if the Connection is not open or if not online.

Set TCPClosedResponse *Value "String"*

Value must be a valid Connection Number (Socket)

If the *Connection* closes, the "*String*" will be inserted into the input stream. This allows scripts to monitor a Connection by using a WaitFor or Trigger command. Sets the Failed flag if the Connection is not open or if not online.

Set TCPEOLTranslation *Value Value1*

Value can be from 0 to 255
Value1 is optional and can be from 0 to 255

Normally the EOL character will be sent “as is” to the active Connection. By changing its value, you can translate this to a specific character required by a particular system. You might wish to change a Return to a Return/Line Feed pair, or a Return to a Line Feed.

Set TCPFlushChar *Value*

Value can be from 0 to 255

Sets the character that is used to trigger flushing the output buffer to the active Connection for modes 1 and 3. This defaults to a Return.

Set TCPFlushFrequency *Value*

Value can be from 0 to 512

The output buffer can hold a maximum of 512 characters. You can change this value to flush data more frequently to the active Connection. Note that in Telnet mode 2, the buffer is flushed as each character is sent.

Set TCPMessages *State*

State can be Off, On

If connecting manually, you can turn off the display of the various progress messages shown by TCP/IP.

Set TCPNotify *State*

State can be Off, On

Normally a message is displayed on screen and put into the capture buffer when a service has been manually switched. This turns on and off the display of that message.

Set TCPReconnect *State*

State can be Off, On

Controls whether TCP/IP attempts to reconnect or not when a Connection is started.

Get TCPServiceEntry *Value VarName*

Value can be from 1 to size of Service list
Value can also be an entry name from the list

Extracts the “dotted address” or “host.name” of the specified Service list entry and stores it in the specified variable. If the entry does not exist, then the Failed will be set. If a name was specified, the match will respect the current state of the *CaseSensitive* flag.

Script and Program Control

Run "*FoldernameFilename*" *Label*

Label is optional

Resets many script parameters then runs the specified script. If *Label* is given control passes to the specified label in the new script. If the label does not exist then a “label not found” error occurs.

Chain "*FoldernameFilename*" *Label*

Label is optional

Passes control (or “Chains”) to the specified script, preserving all the settings that are normally reset when a script is Run. If *Label* is given then control is passed to the specified label in the new script. If the label does not exist then a “label not found” error occurs. *REMINDER: Chaining to a script does not preserve the current For/Next loops, nor does it preserve the “Gosub” stack.*

Chainbacks can now be nested to 16 levels. This allows much more flexible use of the “Chain’ command, as using shorter scripts will always speed script execution.

Chainback

Loads the script that chained to the currently-running script (*$LastScriptPath$LastScriptFile*) and resumes executing script commands at the statement after the Chain command. If the currently-running script was not chained to, then Chainback stops the script.

POP Chainback

POP All Chainback

These commands “pop” or remove one or all of the Chainback levels.

Launch / Exit / Quit

These commands now set prefix 0 and 8 to the folder of the application being launched, which resolves path problems when launching some ProDOS 8 programs.

Quit2

This command works exactly like the “Quit” command, except it does not hangup first.

Variables

CompareStrings "*String1*" "*String2*" *Varname*

Compares the two strings using the Toolbox's `_CompareStrings` command, which is used when sorting most lists (filenames, etc.). The compare is always case *in*sensitive.

Make CaseChange "*String*" *Value Varname*

Value can be a number 1-4

Depending on the *Value*, makes a change to the case of *String* and returns the results in *Varname*. The valid *Values* are:

1 = UPPERCASE
2 = lowercase
3 = First letter of each sentence is capitalized. Like this.
4 = Capitalize Each Word

Store Variables

Shortcut: **Store Vars**

Stores the current values of the ten numbered variables (0 through 9). It does not store any named variables.

Restore Variables

Shortcut: **Restore Vars**

Restores the ten numbered variables (0 through 9) to the previously-stored values. It has no effect on any named variables.

Clear Variables *Varname1*, *Varname2* … *Varname#*

Shortcut: **Clear Vars**

Varnames are optional

Clears the specified variable names to "". If no variable names are specified, the ten numbered variables (0 through 9) are cleared to "".

Delete Variables *Varname1*, *Varname2* … *Varname#*

Shortcut: **Delete Vars** *Varname1*, *Varname2* … *Varname#*

Deletes the specified variable name(s) from the internal table of variable names.

Match String "*TheLine*" "*String1*" … "*String#*"

Looks to see if *String1* exists anywhere within *TheLine*. If so then $Matched is set to “1”, $MatchString is set to *String1*, and the Failed flag is cleared. If *String1* is not found within *TheLine* then the other strings are checked (the only limit on the number of strings is how many can fit into the script expansion buffer).

If no match is found then $Matched is set to “0”, $MatchString is set to "", and the Failed flag is set.

Strip Spaces "*TheString*" *Varname*

Deletes all space characters from *TheString* and stores the results in *Varname*.

Trim Spaces "*TheString*" *Varname*

Deletes only the leading and trailing spaces from *TheString* and stores the results in *Varname*.

Overlay String "*String1*" "*String2*" *Start Varname*

Start can be from 1 to 256

Overlays *String2* on top of *String1*, starting at character position *Start*, and stores the result string into the specified *Varname*. The Failed flag is set if *String2* cannot be overlaid into the existing space used by *String1*.

For example:

```
Overlay String "12345" "ABC" 1 9
# var 9 is now "ABC45"
Overlay String "12345" "ABC" 3 9
# var 9 is now "12ABC"
Overlay String "12345" "ABC" 4 9
# the Failed flag is set because "ABC" cannot be overlaid onto "12345" starting at position 4
   (var 9 is set to "12345")
```

Insert String "*Original*" "*New*" *Position Varname*

Position can be from 1 to 256

Inserts the *New* string into the *Original* string at character *Position*. If the combined string exceeds 256 characters the result is truncated to 256 characters and the Failed flag is set.

Example:

```
# Loop
Display "Insert where? "; Input Line Position
Insert String "****" "NEW" $Position Result
Display "Result is '$Result'.^M"; Goto Loop
# the results:
0 - script error; run the script again
1 - NEW****
2 - *NEW***
3 - **NEW**
4 - ***NEW*
5 - ****NEW
6 through 256 - ****NEW
257 or higher - script error
```

Delete String "*TheString*" *Start Length Varname*

Start can be from 1 to 256
Length can be from 1 to 256

Deletes *Length* characters from *TheString*, starting at position *Start*. If *Start* is 0 or *Start* is greater than the length of *TheString* the Failed flag is set and *Varname* is set to *TheString*.

Example:

```
Delete String "12345" 1 3 9
# var 9's now "45"
Delete String "12345" 3 3 9
# var 9's now "12"
Delete String "12345" 3 256 9
# var 9's now "12"
```

Evaluate "*Expression*" *Varname*

Evaluates the given string expression and sets *Varname* to the result. Valid operators are ( ) + – * / ^ ÷ and – (the last two operators are Option-/ and Option-Minus, respectively). *NOTE: To use the "power of" operator (^) you must type it twice (^^).*

The calculations are made on whole Integer numbers, so any remainders from divisions are lost, with the number being rounded down.

The Failed flag is set if *Expression* is not a valid math expression (i.e. there are characters other than numbers, operators or spaces, the parentheses are not balanced, etc.). The Failed flag is also set if any intermediate calculation overflows 32 bits (4,294,967,296), or if parsing the expression overflows the internal 256-byte parsing buffer.

Finally, the Failed flag is set if the result is a negative number, because negative numbers are not valid elsewhere in Spectrum's scripting language. However, in this case the result *Varname* will contain the positive answer (i.e. if an expression is calculated to be –5 then the Failed flag will be set and *Varname* will be set to 5).

Example:

```
Evaluate "2+(3^^4)-2+(3^^2)-3*4+(6/3+(12/4))" Result
Display "The result is $Result.^M"
Evaluate "2^^$Day" Result
Display "The result is $Result.^M"
```

Getting Input

Ask1 "*Question*" "*Button1*" *VarName {Value}*
Ask2 "*Question*" "*Button1*" "*Button2*" *VarName {Value}*
Ask3 "*Question*" "*Button1*" "*Button2*" "*Button3*" *VarName {Value}*
Question is a string up to 68 characters long
Button# is a string up to 12 characters long
Value is a number 0 to 65535

Version 2.0: Instead of the question being restricted to 68 characters long, these commands now accept questions up to 255 characters long (the size of the alert window is adjusted automatically so the message will fit).

Version 2.1: Added an optional value that specifies the number of seconds to wait for a response. If the user doesn’t respond within that many seconds, the default button is clicked automatically. If there is no default button, and all the buttons have the same first character, then a SysBeep occurs and the Ask alert will not go away.

Ask1 and Ask2 use the “cool” button layout. Due to formatting problems with Ask3 if the button names are more than a few characters, Ask3 still uses the older button layout.

Waitfor String "*String1*" "*String2*" … "*String7*" "*String8*"

- Comments after the strings were being seen as target strings. Fixed.
- Double letters could cause Waitfor String not to see a string it was waiting for. Fixed.

Set IdleTimer *Value*

Value can be from 0 to 65535

When waiting for input using any “Getting Input” command that gets data from the port, Spectrum monitors the port activity. If the port is idle for *Value* seconds then the command will fail and Spectrum continues executing the script.

This command works in *conjunction* with the Timeout value (Timeout fails the command after *Value* seconds **regardless of port activity**, whereas IdleTimer fails the command after *Value* seconds of **no port activity**).

Example:

```
Set Timeout 120 # 2 minutes
Set IdleTimer 30 # 30 seconds
Transmit "Send me the entire file list!^M"
Waitfor String "END OF LIST"; If NOT Failed Then Display "The entire list was received within 2
   minutes."; Stop Script
# otherwise the Waitfor command failed...
On $FailureCode GotoNext TimedOut, IdledOut
# if here then $FailureCode is 0...
Display "Failed for some other reason (a Hitzone was probably clicked)."; Stop Script
# TimedOut - after 2 minutes there's still port activity
Display "Still receiving the list."; Stop Script
# IdledOut - gone 30 seconds with no port activity
Display "Either the list is complete but something happened that the 'end of list' string was not
   seen, OR the host is disconnected/lost in space."; Stop Script
```

Read Char *Varname*

Accepts one character from the port and stores it into the specified variable.

If a timeout was used and time ran out, the Failed flag is set.

Read Line *Varname*

Accepts up to 256 characters from the port. When a Return character is received the line is stored into the specified variable.

If a timeout was used and time ran out, the Failed flag is set and the specified variable contains the data that was read so far.

Branching and Loops

IMPORTANT: The script keyword "GotoNext" cannot be separated as "Goto Next" (version 1 was more forgiving of this error; version 2 will instead attempt to GOTO a label named "Next").

On CarrierLoss Goto *Label*
On CarrierLoss GotoNext *Label*

If this command is active and Spectrum detects the transition from having a carrier to having no carrier, then control will be passed to the specified label in the currently-running script.

This command does nothing unless "DCD Is Valid" is checked, the modem has been initialized properly, and the modem is connected with a properly-wired modem cable.

On CompareStrings "*String1***" "***String2***" Goto** *LessLabel***,** *EqualLabel***,** *MoreLabel*
On CompareStrings "*String1***" "***String2***" GotoNext** *LessLabel***,** *EqualLabel***,** *MoreLabel*

Compares the two strings using the Toolbox's `_CompareStrings` command, which is used when sorting most lists (filenames, etc.). Based on the result of the comparison, script control jumps to *LessLabel*, *EqualLabel*, or *MoreLabel*.

Set Labels *State*
State can be Off or On

Works as before, but adds support for an extended keyboard: Pressing function keys 1 through 15 will attempt to "Gosub" to that label.

On *Value* **Goto** *Label1*, *Label2* … *Label#*
On *Value* **GotoNext** *Label1*, *Label2* … *Label#*
On *Value* **Gosub** *Label1*, *Label2* … *Label#*
On *Value* **GosubNext** *Label1*, *Label2* … *Label#*

These commands have been modified to allow as many labels as will fit in the script line buffer (currently 636 characters, minus the number of characters used for the command itself).

If Value is 0 or is greater than the number of labels supplied, the Failed flag is set and control continues to the next statement. For example, the Display statement would be executed in this script:

```
On 5 Goto One, Two; Display "Didn't go anywhere"; Stop Script
```

For *LoopNumber Start Stop Step*

Works like version 1, except if the initial *Start* value is less than the initial *Stop* value the loop counts.

Example:

```
Display "1 to 100 by twos:^M^J"
For 0 1 100 2; Display "$ForValue0^I"; Next 0
Display "^M^J^M^J100 to 1 by twos:^M^J"
For 0 100 1 2; Display "$ForValue0^I"; Next 0
```

On Compare *Value1 Value2* <u>GoXXX</u> *LessLabel*, *EqualLabel*, *MoreLabel*

<u>GoXXX</u> can be Goto, GotoNext, Gosub, or GosubNext

This command is similar to the Compare command, except that the result can be acted on immediately (an intermediate variable to hold the result is not required):

If *Value1* < *Value2* then control passes to *LessLabel*

If *Value1* = *Value2* then control passes to *EqualLabel*

If *Value1* > *Value2* then control passes to *MoreLabel*

Conditional Tests

If False *Varname* **Then** *Statement*

If *Varname* is empty ("") or is “0” then *Statement* is executed.

If True *Varname* **Then** *Statement*

If *Varname* is not empty ("") and is not “0” then *Statement* is executed.

If Not Contains "*ShortString***" "***LongString***" Then** *Statement*

If *ShortString* is longer than *LongString* then *Statement* is now executed.

If Null *Varname* **Then** *Statement*
If Null "*String***" Then** *Statement*
If Null *$EditorHandle#* **Then** *Statement*

If a variable name is supplied and the variable is empty ("") then *Statement* is executed.

If a string is supplied (the current quote delimiters are required) and the string is empty ("") then *Statement* is executed.

If an $EditorHandle# replacement item is supplied and the specified editor is empty ($EditorSize# is 0) then *Statement* is executed.

If Not Null *Varname* **Then** *Statement*
If Not Null "*String***" Then** *Statement*
If Not Null *$EditorHandle#* **Then** *Statement*

If a variable name is supplied and the variable is **not** empty then *Statement* is executed.

If a string is supplied (the current quote delimiters are required) and the string is **not** empty then *Statement* is executed.

If an $EditorHandle# replacement item is supplied and the specified editor is **not** empty then *Statement* is executed.

If Defined *Varname* **Then** *Statement*

If the specified variable name has been defined then *Statement* is executed.

If Not Defined *Varname* **Then** *Statement*

If the specified variable name has **not** been defined then *Statement* is executed.

If Desktop Then *Statement*

If the 640 mode SHR display with menu bar is showing then *Statement* is executed.

If Not Desktop Then *Statement*

If the 640 mode SHR display with menu bar is **not** showing then *Statement* is executed.

Else *Statement*

All “if” comparisons set a special flag that remembers whether the “if” statement was true or false. The “else” command tests this flag and executes the rest of the line if the previous “if” statement was false. *NOTE: “Else” is a completely separate command; it is* ***not*** *an optional part of an “if” statement!*

Using “else” clears the special flag, so multiple “else” statements do not work.

Example:

```
If Equal "A" "B" Then Display "EQUAL^M^J"
ELSE Display "NOT equal^M^J"
ELSE Display "This will never been displayed.^M^J"
# change "B" to "A" and run again
# also try using "If NOT Equal" and other conditional tests
```

Screen Appearance

Set Flush *State*

If Flush was On in version 1, processing incoming data had greater priority over executing the script commands. In version 2 both functions are approximately equal priority, so script commands will execute regularly even if lots of incoming data is pending.

Set ScreenBlank *State*

State can be Off, On, or Auto

Spectrum now completely controls the blanking (previously if Twilight II was active Spectrum would ask it to blank the screen). Also, the border color is now set to black when blanked.

Set ScreenBlank OFF now just broadcasts the "systemSaysForceUndim" IPC message, so if Twilight II (for example) blanks in the background, your script can include a Set ScreenBlank OFF command to force the screen to be unblanked. *NOTE: It **is** acceptable to issue "Set Screen OFF" even if the screen is already unblanked.*

Set ScreenBypass *Value*

Value can be On or Off

Controls whether data is sent to the screen or not. If set to 'Off', it allows script execution to continue with a closed screen. Note that this is not the same as "Set Screen Off" as the "WaitFor" and "Trigger" commands will still operate. Effectively it operates as though a screen was open, but with no data showing on screen.

Set XferStatus *Value*

Value can be On or Off

Controls whether the Protocol Transfer Status box is shown during a file transfer.

Capture Buffer Control

Load Buffer "*FoldernameFilename*"

- This command would overrun memory if the file was too big to be loaded. Fixed.

Append CaptureFile "*FoldernameFilename*"

Opens the specified text file and begins capturing incoming data to the end of it.

What's new: If the specified file does not exist it will be created automatically. If the specified file is not a text file then a script error is generated.

Set AutoSave "*FoldernameFilename*"

If AutoSave is specified, and AutoSaveBuffer is on, and Append is on, whenever the capture buffer fills it will be automatically appended to the AutoSave folder and filename.

In version 1, if the AutoSave folder or filename was invalid, Spectrum would not save the buffer. In version 2 Spectrum gives the user the opportunity to manually Clear, Save, or Append it.

Transferring Files

Receive File *Protocol*

- When receiving Zmodem, if the host did not start the transfer, or aborted the transfer before Spectrum's file transfer status window appeared, the script would pause for a very long time. Fixed.

Send File ZBatch *"File1" "File2" ... "File3"*

Send File ZBatch *ScriptEditor*

Send File YBatch *"File1" "File2" ... "File3"*

Send File YBatch *ScriptEditor*

File is the path to a file or a file in the Send file folder
ScriptEditor is a value from 0-9

Batch sends multiple files. These commands replace the functionality of the BatchXfer XCMD.

OS Utilities

Rename "*Foldername1Filename1*" "*Foldername1Filename2*"

- Renaming “FILENAME” to “Filename” would raise an error. Now the case is changed without generating an error.

Copy File "*Foldername1Filename1*" "*Foldername2Filename2*"

Can now copy even very large (“tree”) files, where previously it showed an “access not allowed” error.

- Now handles Macintosh-style OptionLists.
- Now preserves original file creation date and time.

Get FileInfo "*FoldernameFilename*" *Varname*

Works as documented, with the following additions to the returned string:

Start	Length	Information
59	6	Creation date in the format YYMMDD
65	4	Creation time in the format HHMM
69	6	Modification date in the format YYMMDD
75	4	Modification time in the format HHMM

Get FileInfo2 *"FoldernameFilename" VarName*

Complements the *"Get FileInfo"* command, but returns the LongYear instead of the two digit short year. This makes the command Y2K compliant.

Set FileInfo "*FoldernameFilename***" "***TTIAAAA***"**

TT is a hexadecimal filetype 00-FF
AAAA is an optional hexadecimal auxtype 0000-FFFF

Sets the specified file's filetype and auxtype. The auxtype (AAAA) can be omitted and only the filetype will be changed.

Get MacFileInfo "*FoldernameFilename***"** *Varname*

Sets *Varname* to contain the Macintosh file's Type and Creator. The Failed flag will be set if the file is not an extended file.

NOTE: If the file has a "pdos" creator, Get MacFileInfo may return the filetype holding ASCII values that are not printable. This is because with a creator of "pdos" you can have the filetype and auxtype embedded as Hex data in the four characters of the Mac filetype.

Set MacFileInfo "*FoldernameFilename***" "***TypeCrtr***"**

If the specified file is a Macintosh file then this command will set the file's Type and Creator information.

Show File "*FoldernameFilename*" *Start Stop*

Start and *Stop* are optional

As in version 1, this command displays the file to the screen (and outputs it to the port if the Echo option is on). What's new:

- The optional *Start* and *Stop* values let you show a specific portion of the file.
- While showing a file, the Spacebar will alternately pause and restart the display.

Get File "*PromptString*" *Kind Varname*

Works as it did in version 1, but Kind can now also be a 3 or 4.

When using a *Kind* of 3 the dialog box displays launchable applications and also contains two checkboxes that let the user control whether the script should hang up before launching ($DoHangup), and whether quitting the next application should return to Spectrum ($DoReturn).

When using a *Kind* of 4 the dialog box displays Teach and text files only (not Appleworks Classic).

Copy File *"SourcePath" "TargetPath" Value*

Value is optional

```
Value = 1 copies data fork
Value = 2 copies resource fork
Value = 3 copies both forks
```

Using the optional *Value*, you can copy either the data fork, resource fork, or both forks. In all cases, the target file must already exist, or an error will be generated. The resource fork will be created if needed.

JudgeName *"FoldernameFilename" VarName*

Expands the *FoldernameFilename* and parses it to suit the FST of the destination volume pointed at by *Foldername*.

Get Multifile *"PromptString" Value ScriptEditor*

Value = 0	Any file
Value = 1	Text only
Value = 2	Text, AppleWorks, Teach
Value = 3	Launchable Applications
Value = 4	Text and Teach

ScriptEditor is a value from 0-9

This complements the “Get File” command. It is primarily intended for use by the “Send Batch” commands.

Reading and Writing Files

NOTE: Scripts can now open 10 files and 10 catalogs (previous limits were 4 files and 4 catalogs).

Read File *FileNumber Varname NumBytes*

FileNumber can be from 0 to 9
NumBytes is optional; if used it is a value from 1-256

If the optional *NumBytes* parameter is not provided then this command behaves exactly as it did in version 1.

If *NumBytes* is specified then this command reads exactly that many bytes from the file (it does not check for Return). If reading *NumBytes* passes the end of the file, the Failed flag is set and the variable contains the bytes that were read successfully.

Write File *FileNumber* **"***String***"** *NumBytes*

FileNumber can be from 0 to 9
NumBytes is optional; if used it is a value from 1-256

If *NumBytes* is not specified then all the characters of the *String* are written; if *NumBytes* is specified then only the first *NumBytes* characters of the *String* are written.

Set FilePos *FileNumber Position*

FileNumber can be from 0 to 9

After opening a file you can use this command to quickly move the file marker to a specific character position. For example, **Set FilePos 0 83** will set the file marker to the 83rd character in file 0.

If you set to a character position that is past the end of a file, the file length will not get extended until you begin writing to that position.

For example, this script…

```
Open File 0 ":Ram5:NewFile"
Set FilePos 0 10000
Set FilePos 0 10; Write File 0 "Word.^M"
Close File 0
```

…will result in a file that is only 15 characters long, not 10,000. To extend a file Spectrum writes as many Return characters as necessary (in the example above, Return characters are written to position 1 through 9, so the “W” of “Word” will start at the character position that was set).

Search File *FileNumber* **"***String***"**

FileNumber can be from 0 to 9

Searches the given file for the given string. The search starts at the current file position and proceeds toward the end of the file.

If *String* is found then the Failed flag is cleared and the file marker is set to the first character *after* the string. If *String* is not found then the Failed flag is set.

Reading Catalogs

Read Catalog *CatalogNumber Varname*

The information that is returned has been extended—see “Get FileInfo” in the OS Utilities section.

Read Catalog2 *CatalogNumber VarName*

Complements the “Read Catalog” command, but returns the LongYear instead of the two digit short year. This makes the command Y2K compliant.

Script Editor

All script editor commands have been modified to allow an optional *EditorNumber*, a value from 0-9. If *EditorNumber* is not present, editor number 0 is assumed:

Clear ScriptEditor *EditorNumber*

Send ScriptEditor *EditorNumber*

Print ScriptEditor *EditorNumber*

Apply Replace *EditorNumber* "*FindString*" "*ReplaceString*" *Varname*

Apply LowASCII *EditorNumber*

Apply RemoveControls *EditorNumber*

Apply LFsToCRs *EditorNumber*

Apply RemoveSpaces *EditorNumber*

Apply Special *EditorNumber Value*

Apply Format *EditorNumber Value*

Append ScriptEditor *EditorNumber* **"***Item***"** *Kind*
Load ScriptEditor *EditorNumber* **"***Item***"** *Kind*
EditorNumber is optional; if used it is a value 0-9
Item can be a FoldernameFilename, or can be ::Scrollback, ::Buffer, or ::Clipboard
Kind is an optional Varname

Loads the specified data into the given ScriptEditor (Load replaces the script editor contents, while Append adds the new data to the end of the specified script editor).

If *Item* is a file on disk, an optional *Kind* varname can be used which will be set to indicate what kind of file was loaded: 1=Text, 2=Teach, 3=AppleWorks Classic.

- This command can now load Macintosh text files, even if the file has a Macintosh resource fork.
- If a Teach file was loaded, then a Text file was loaded, the ScriptEditor still thought it had a Teach file in it. Fixed.

Save ScriptEditor *EditorNumber* **"***FoldernameFilename***"** *Kind*
EditorNumber is optional; if used it is a value 0-9
Kind is optional; 1 saves as a Text file; 2 saves as a Teach file; 3 saves as an AppleWorks Classic file

Saves the specified scripteditor to disk. If *Kind* is not specified then the format is Text (if a text or AppleWorks Classic file was loaded) or Teach (if a Teach file was loaded). Otherwise the *Kind* value forces a specific file type to be saved (only Teach saves any style information; the other formats only save the text from the specified script editor).

Create ScriptEditor *EditorNumber*

Creates an empty scripteditor. If the scripteditor is already in use then the Failed flag is set and the existing scripteditor is not changed.

Show ScriptEditor *EditorNumber*

Displays the specified scripteditor on the screen. Holding down the Option key pauses the display; pressing Escape cancels the display.

Clear SEDirty *EditorNumber*

Clears the “dirty” flag for the specified editor.

Error Control

ResumeNext

Use only in an “On Error Goto” procedure

Continues running the script *after* the command that caused the error. *NOTE: This can be confusing; it is intended for advanced script authors.*

Script Interpretation

Set Quote *Character*

A script error could occur when the quote character was set to something other than the standard character (") and a line contained a semicolon:

```
Set Quote %
Display %This is fine.%
Display %This ; failed.%
```

This has been fixed (i.e. the second Display command will no longer fail).

Set Replacement *Character*

Changes the value normally used for the “$” substitution character.

Advanced Commands

Store Screen

Saves the current screen characters to an internal buffer. The Failed flag is set if a display is not open, or if the opened display does not support the feature.

Restore Screen

Restores the screen characters from the previously-stored screen. *NOTE: This command closes and re-opens the Online Display for a proper update to occur. This is quick, but does cause a flicker on the screen.*

The Failed flag is set if a display is not open, or if the opened display does not support the feature.

Define Trigger *Value* "*String*"

Value can be a number 1 to 16
String can be up to 256 characters

Defines a sequence of characters that Spectrum should watch for whenever it is processing incoming data. If *String* is "" then the trigger is deleted, just as Delete Trigger

If the character sequence is seen and an “On Trigger Goto…” command is active, control passes to the specified *Label*.

Note that triggers take precedence over the active command. For example, if “Waitfor String” is currently waiting for “password” and a trigger has been defined to watch for “sword”, as soon as the “d” arrives the trigger causes the “Waitfor String” command to fail (the Failed flag will be set) and control will pass to the trigger handler.

Get TriggerInfo *Value Varname*
Value can be a number 1 to 16

Sets *Varname* to the currently-defined trigger string.

Delete Trigger *Value1*, *Value2*, … *Value#*
each *Value* can be a number 1 to 16

Deletes the specified trigger number so its associated trigger string is no longer searched for.

Delete Triggers
Deletes all the triggers so no trigger is searched for.

On Trigger Goto *Label*
On Trigger GotoNext *Label*
When a defined trigger is seen, control jumps to the specified *Label* in the currently-active script, and the *$Trigger* replacement item identifies which trigger was seen. As with other “On X Goto” commands, “Resume” and “ResumeNext” will resume script execution where it was interrupted.

At the entry to *Label*, if the Failed flag is set then a script command was interrupted when the trigger was seen. If the Failed flag is clear then the trigger was seen while Spectrum was flushing data to the screen.

Open ResFile *FileNum* "*FoldernameFilename*"
FileNum can be from 0 to 9

Opens the resource fork of the specified file, which adds it to Spectrum’s resource search path. The Failed flag is set if the specified file’s resource fork could not be opened.

If a script frequently uses commands that access resources out of a particular file (such as Draw Picture and Draw Icon) the script can be sped up by opening the resource file in advance. This is much faster because Spectrum doesn’t have to open and close the resource file around each command; Spectrum will instead just always find the resource in memory (the most-recently opened resource file is searched first).

This command lets scripts create and store all its related resources in one file. However, because these resources are opened and installed at the head of Spectrum's resource search path, script authors who use this feature must take care not to use resource ID numbers that conflict with any of Spectrum's resources (otherwise Strange Things may happen).

When creating a resource file for use with Spectrum, use resource IDs in the range $7001 and above to avoid conflicts. *NOTE: Resource names are case sensitive.*

Close ResFile *FileNum*

FileNum can be from 0 to 9

Closes the specified resource file.

Close ResFiles

Closes all resource files opened with the Open ResFile command.

Set Failed

Sets the Failed flag.

Clear Failed

Clears the Failed flag.

Shut Down

Closes any files opened by the script, calls the AutoSaveBuffer routine, stops the script, then shuts down the system.

Set DiskErrors *State*

State can be Off or On

When a disk-related error occurs (e.g. “file busy” or “disk full”) Spectrum normally generates a script error. The script error can be caught and handled by the script using an On Error Goto routine.

Another approach is to turn DiskErrors off, which will not generate a script error for disk-related problems. Instead, when a command that accesses the disk is used, the Failed flag is set or cleared to indicate whether the command was successful.

If the disk command fails a script can use the $ErrorCode and $ErrorMsg replacement items to determine what error occurred.

NOTE: Only disk access commands are affected; script errors (e.g., using a file number that is already in use) will still cause a script error to appear.

Set ScriptLock *State*

State can be Off or On

ScriptLock can be turned on to prevent a script from being interrupted by the user. When ScriptLock is on:

- On the File menu: the Launch and Quit menu items are dimmed, and the Close menu item is dimmed when the Online Display is frontmost.
- On the Phone menu: the Answer Back menu item is dimmed.
- The entire Script menu is dimmed.
- Pressing Escape will not stop the script.

If you use this command you must be very careful that you provide some way to exit your script, otherwise the user will have to restart the computer to quit Spectrum!

- Previously, ScriptLock On would completely ignore the Escape key. Now the Escape key is processed normally if "On Escape" is active.

Compress Script "*Source*" "*Destination*"

Source is the *FoldernameFilename* of a Spectrum Script file
Destination is a *FoldernameFilename* parameter

Compresses the data fork of the *Source* file and saves it as *Destination* with the “Compressed Spectrum Script” file/auxtype. A script error occurs if an error occurs. This command is intended only to compress script files, so it works only on Teach or Text files whose data forks are less than 64K.

There is no corresponding Uncompress command—the only way to use a compressed script file is by choosing Run A Script from the Script menu, by running a script that RUNs or CHAINs to the script, or by double-clicking a compressed script in the Finder.

When compressing a script you should be sure to keep the *Source* file intact. To help prevent mistakes, *Destination* must not already exist, otherwise the Failed flag is set.

Save Editor

If the editor contents need to be saved, this command saves the contents. If the editor had not be saved before (i.e. it is a new file) then the command presents the standard file dialog box so the user can specify where to put the file and what to name it.

The Failed flag is set or cleared to indicate the success of the command.

External *CommandName Data*

Shortcut: **Ext**
CommandName is required
Data is optional (depends on the particular external)

This broadcasts an IPC message to `Spectrum XCMD~COMMANDNAME~`. The external acts on the command (what the command does, and the format of any *Data* being passed, is entirely up to the external module).

The Failed flag is cleared or set to indicate the success or failure of the command.

Compile Script *"SourcePathFileName" "TargetPathFileName"*

Optimises a script by converting the commands into tokens, and removing any white space and REM statements. This can speed up the execution of some scripts by a factor of two.

Set KeyTranslation *Value*

Value can be “None” or “Standard”

By default, KeyTranslation is set to “None” while the online display is open. This script command overrides that setting.

Set Signature *“String”*

Assigns a “*String*” that will be inserted into any open TextEdit control with the OA-T key command.

Reboot

Similar to “*Shut Down*”, but reboots the system instead of just shutting down.

Specialty Commands

Load Screen "*FoldernameFilename*"

Loads a screen image graphic (packed $C0/0001 or unpacked $C1/0000) into the Spectrum SHR display (Spectrum's Save Screen command creates a suitable file). Only the picture data is loaded; the SCBs and palettes are ignored, as are the first 13 scanlines of the graphic (skipping the menu bar).

This command does nothing if the Spectrum SHR display is not open and frontmost.

NOTE: Loading a screen just loads a graphic image of the screen. Like the Draw commands, screen updates will not refresh the graphic image.

DirectAction "*String*" *Varname*

As in v1.0, this command passes *String* to the current online display for processing; and the results are returned in *Varname*. The options available depend on the online display being used, and in v2.0 the Spectrum SHR display added two new commands. Refer to the online help for details.

Set Update

Sets the $Update replacement item to 1. If a script is watching the $Update replacement to see if a screen update is needed, this will make the script think Spectrum has just redrawn the display in response to a system update event.

Clear Update

Sets the $Update replacement item to 0. When a screen update occurs on a super hires display, $Update is set to 1, which is a signal to advanced scripts to redraw special screen elements (rectangles, icons, etc.). After the script updates the screen it would include a Clear Update command so the screen won't be updated again until another update is needed.

Draw Rectangle *Left Right Top Bottom Fill Frame Radius*

Works as it did in version 1, with the addition of the optional *Radius* value (1-64) to create rectangles with rounded corners.

Draw Icon *X*,*Y* "*Icon*" "*FoldernameFilename*"

Works as it did in version 1, except if the specified resource is not found the Failed flag is set (no script error).

Draw DimIcon *X*,*Y* "*Icon*" "*FoldernameFilename*"

This command is identical to the Draw Icon command, except that it draws the icon so it looks "dimmed."

Draw Picture *X*,*Y* "*Picture*" "*FoldernameFilename*"

Works as it did in version 1, except if the specified resource is not found the Failed flag is set (no script error).

Draw Text *X*,*Y* "*String*" "*Family*" *Size Style Color*

X is a value from 0 to 639
Y is a value from 0 to 186
Family is optional; if used it is a either value or a string that identifies the font
Size is optional; if used it is a value from 2 to 255
Style is optional; if used it is a value from 0 to 31
Color is optional; if used it is a value from 0 to 15

Draws *String* on the screen at coordinates (*X*,*Y*). The standard font is Shaston, 8 point, plain text, in black. If *Family* is specified then that font is used (if available in the system). If *Size* is specified then that size is used, and so on.

To determine the correct *Style* value, add the values for each style you want to use:

Plain = 0	Bold = 1	Italic = 2
Underline = 4	Outline = 8	Shadow = 16

For example, to use Bold+Italic+Shadow the *Style* value would be 19 (1+2+16).

Define Hitzone *Num Left*, *Right*, *Top*, *Bottom CornerRadius Color*

Num is a value from 1 to 32
Left and *Right* are values from 0 to 640
Top and *Bottom* are values from 0 to 186
CornerRadius is optional; if used it is a number from 0-64
Color is optional; if used it is a value from 0 to 15

Defines a rectangular area that can act like a "button." *NOTE: This command works only on 640 mode super-hires screen displays.*

When hitzones are turned on, clicking inside a hitzone will highlight it. If the mouse button is released while inside a hitzone the $Hit replacement item will indicate which hitzone number was clicked.

Clear Hit

If $Hit is not zero then a hitzone was clicked. After performing the desired action your script must include a Clear Hit command to reset $Hit to 0 (otherwise your script will think that the user has clicked a hitzone again).

Flash Hitzone *Num*

Num is a value from 1 to 32

Simulates a click on the specified hitzone, exactly as if the user had manually clicked it (i.e., $Hit is set to the given number and any associated HitAction is performed). The Failed flag is set if the specified hitzone does not exist, or if the SHR display is not open.

Delete Hitzone *Num*

Num is a value from 1 to 32

Erases the frame around the specified hitzone, then deletes the hitzone so it is no longer active.

Delete Hitzones

Deletes all the hitzones, but does not erase each individual hitzone (Display “^L” to clear the screen).

Set Hitzones *State*

State can be Off or On

Determines whether hitzones are active or not (when Off, clicking inside a hitzone does nothing).

Set HitAction *Action*

Action is a value from 0-4

After the mouse button goes down in a hitzone the zone is highlighted and tracked. If the mouse button is released while still inside the hitzone then Spectrum sets the $Hit replacement item to the hitzone number that was clicked. Optionally, an additional action can occur:

Action	Result
0	Do nothing else
1	Post an ESCape keypress (will trigger on “On Escape” handler, or will abort the script if On Escape has not been used)
2	Post a Return keypress (will complete a Get Key, Get Line, Input Key, Input Line, or Waitfor Keyboard command)
3	Post a Return into the port input stream (will complete a Get Key, Get Line, Read Char, or Read Line command)

4	Post “Hitzone” text into the port input stream (will complete a Waitfor String “Hitzone” command)

Store Hitzones

Stores the current hitzones and the associated settings.

Restore Hitzones

Restores the previously-stored hitzones and settings and redraws the hitzones on the screen.

Set Hit *Num*

Num is a value from 0 to 65535

Sets the $Hit replacement to the specified number. This is mainly available for XCMDs to use, but could be useful if your script wants to “fake” a hit.

Set Cursor *Value*

Value is a value 0-6

Sets the mouse cursor to one of the following pointers:

#	Cursor Action
0	Hide (hides the current cursor)
1	Show (shows the current cursor)
2	Arrow
3	Watch (animates if a cursor animation extra is installed)
4	Spectrum Watch (won’t animate)
5	I-Beam
6	Forces the Cursor into selected format

Freeze Cursor

Unfreeze Cursor or Melt Cursor

Hides and disables the Cursor. This is the same effect as you will get when downloading files. The cursor will be restored when the script stops.

Drop DTR *Value*

Value is a number from 1-60

Drops the DTR line to the modem for the given number of seconds. Without the optional value, it defaults to 2 seconds.

Post Input *"String"*

Posts the "*String*" into the input buffer as if it had come from the port.

Show Window *"String"*

Opens a dialog style window and displays the *"String"* in LETextBox format. This window complements the new commands for "*Set ScreenBypass*" by allowing messages to be displayed while there is no active window open.

Close Window

Closes the window opened by "Show Window".

Index