Comms

User Guide

CONTENTS

Comms

Comms

INTRODUCTION

This User Guide provides further information about the Series 5's Comms program. Comms provides terminal emulation and file transfer to other computers. You can use it to:

- Connect the Series 5 directly to another computer.
- Connect the Series 5 via a modem to access electronic mail systems and bulletin boards.

Note: You can also transfer files between the Series 5 and a PC using PsiWin 2.x and the Docking cable. This allows file conversion between PC and Series 5 formats. If you do not already have PsiWin 2, contact your nearest Psion distributor as listed in the Psion distributors list supplied in the Series 5 box for further details.

ABOUT COMMS

Comms is the Series 5's communications and terminal emulation program. You can use it to connect to remote computers, control modems and use your Series 5 as a remote terminal when connected to a host computer.

The program has two 'views':

- The Terminal emulation screen, where you connect to and communicate with remote machines. This is the screen you see when you first start Comms.
- The Script editor, where you create 'scripts' to automate parts of communications, e.g. logging on to a remote host.

This User Guide provides an introduction first to the Terminal emulation screen and its features, and then gives a guide to creating your own scripts, together with a glossary of all the commands in the Script language.

STARTING COMMS

To start Comms:

1. Make sure that the Remote link on the Series 5 is switched **off**. To do this, select **'Remote link'** on the **'Tools'** menu in the System screen and ensure 'Off' is displayed in the **'Link'** line.
2. Tap the Extras bar icon, then tap the Comms program icon.

When Comms starts, you will see the Terminal emulation screen, and an **'Online'** information message will appear.

Comms communicates with other machines through either the serial or infrared port at the back of the machine. The message in the status bar will read 'Online' or 'Offline' depending on whether or not a port is active at any given time.

- If no port is currently active, check that the Remote link is off and that the **'Port active'** command on the **'Transfer'** menu is ticked.

Note: To hide the status bar, remove the tick from **'Show status bar'** on the **'View'** menu.

- **To close Comms:** use the **'Close'** command on the **'File'** menu.

Comms

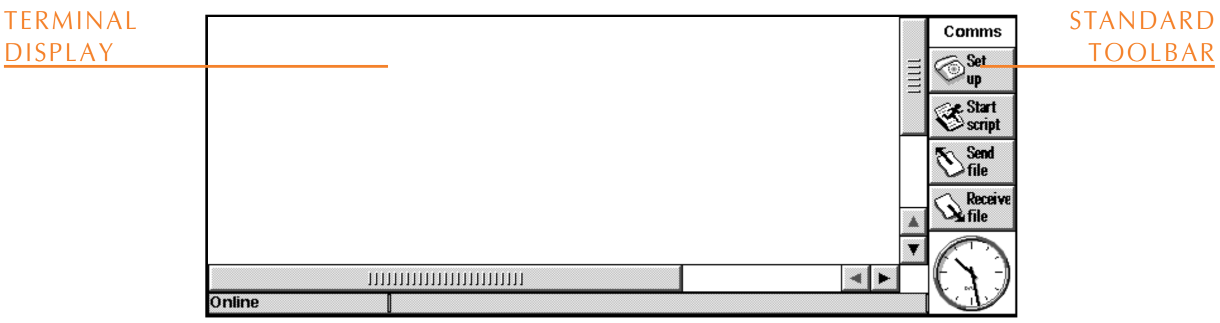
THE TERMINAL EMULATION SCREEN

The Terminal emulation screen is where you:

- Set up the Series 5 for communication with a remote machine.
- Start and stop connections to remote machines.
- Type characters that are sent via the communications port to the remote machine or modem. The characters you type on the Series 5 and those you receive from remote machines are displayed in the Terminal screen.

Note: Displaying the characters you type on the Series 5 is called “local echo”. To switch the local echo off, select **‘Translate codes’** from the **‘Tools’** menu, and remove the tick from the **‘Local echo’** line.

- Send and receive files.



SETTING UP TO COMMUNICATE WITH REMOTE MACHINES

Before starting a connection, you need to make sure that the communication settings on the Series 5 match those of the device you are connecting to. Use **‘Communication settings’** on the **‘Tools’** menu, or the **‘Set up’** button on the Toolbar, to adjust the following communications parameters:

- The comms port used.
- The Baud rate (the speed of data transfer between the two machines).
- The number of data and stop bits.
- Parity.
- The “Handshaking” options.

Comms

COMMS PORT

Set the **'Use comms port'** line to:

- **'Serial port 0':** to connect using the serial port at the back of the machine, e.g. if you are using the Docking cable with an adaptor to connect to a modem.

- **'Infrared':** to start an infrared communications link.

BAUD RATE

The Series 5 can transfer data at speeds of up to 115200 bits per second. Select an appropriate speed in the **'Baud rate'** line of the **'Communication settings'** dialog. Make sure the speed you have set on the Series 5 matches the speed setting on the remote machine, and does not exceed the maximum speed of the modem you are using.

Note: Modern modems can sense and adjust to the speed at which you are transmitting. With older modems, you may need to specify the exact speed of the connection.

DATA BITS, STOP BITS AND PARITY

The data bits, stop bits and parity settings determine how each character to be transmitted is converted into bits.

- Set the number of data bits to the same as the setting on the remote machine.

- Set the number of stop bits to be the same as the setting on the remote machine. Usually 1 stop bit is sufficient, but with low Baud rates (300 or less), or when communicating with some mechanical devices, you may need to set 2 stop bits. Having 2 stop bits set instead of 1 will never cause an error (though the transfer rate will be slightly slower). However, setting 1 when you should have 2 will cause errors.

- You can choose from none, even and odd parity. Make sure the setting is the same as the one on the remote machine.

HANDSHAKING

Use the tick boxes on the **'Handshaking'** page of the **'Communication settings'** dialog to select which of the following are used when communicating:

- XON/XOFF

- RTS/CTS

- DSR/DTR

- DCD

When communicating via a modem, you should normally set XON/XOFF, DSR/DTR and RTS/CTS handshaking, though this may vary according to the remote setup and your modem manufacturer's guidelines.

- Handshaking is sometimes termed 'flow control'.

- For 'Hardware' flow control, tick the 'RTS/CTS' box.

THE TYPE OF TERMINAL EMULATION

Comms can emulate a plain "teletype" or a VT100 terminal.

- **To use one of these:** select **'Terminal emulation'** on the **'Tools'** menu. Choose 'TTY' for a teletype, or 'VT100' in the **'Terminal to emulate'** line.

You can adjust the width and height of the screen by setting the number of characters for each in the Terminal emulation dialog. If you set a greater width or height than can be displayed in the current zoom state, you can use the scroll bars to view other areas of the 'screen'.

Note: To hide the scroll bars, remove the tick from **'Show scroll bars'** on the **'View'** menu.

- Use **'Clear screen'** on the **'Edit'** menu to remove all the characters from the display.

SPECIAL CODES

Different computers may require different codes to be sent when the Enter key is pressed. Comms is initially set to add a "line feed" characer to both incoming and outgoing "carriage returns".

- **To change this:** select **'Translate codes'** on the **'Tools'** menu, then use the **'Add LF to incoming CR'** and **'Add LF to outgoing CR'** lines.

CONNECTING TO ANOTHER MACHINE

There are three ways you can connect your Series 5 to a remote machine using Comms:

- Direct connection to another machine.
- Connection to another machine via a modem link.
- Connection via infrared.

DIRECT CONNECTION TO ANOTHER MACHINE

To connect the Series 5 directly to another machine, e.g. a PC:

1. Connect the Docking cable to the serial port at the rear of the Series 5, and to a COM port on the PC. You may already have this set up if you use PsiWin 2.x. Note the COM port on the PC that the cable is connected to.
2. Make sure neither machine is running other software that uses the communications ports. In the case of the PC, ensure you are not currently running PsiWin 2.x. Make sure the Series 5 has the Remote link switched off, and is not trying to carry out an infrared transfer.
3. Start Comms on the Series 5, and run a suitable Terminal application on the other machine. If you use Windows 95, this may be HyperTerminal.
4. Set both pieces of software to use the appropriate ports, and to use identical settings for Baud rate, Data bits, Stop bits, Parity and Handshaking. The two machines are now ready to exchange characters and files.

Comms

SETTING UP HYPERTERMINAL FOR CONNECTION TO A SERIES 5

Note: The first time you use HyperTerminal to connect to the Series 5, you will need to create a "new connection". To do this:

1. Select **'HyperTerminal'** from the **'Programs|Accessories'** branch of the **'Start'** menu, and double-click on the '**Hypertrm'** icon. Select an icon and give a name to the connection, e.g. 'My Series 5'.
2. In the **'Phone number'** dialog, make sure the **'Connect using'** line is set to the COM port to which you have connected the Docking cable.
3. Use the **'Properties'** dialog to specify identical communication settings to those on the Series 5. In the **'Flow control'** line, **'Hardware'** is equivalent to having the **'RTS/CTS'** box ticked on the handshaking page of the **'Communication settings'** dialog in Comms.

WHEN THE CONNECTION HAS BEEN ESTABLISHED

Once you have established a connection, you will see the characters you type on the Series 5 appear in the PC's display, and characters typed on the PC will appear in the Comms Terminal emulation screen. This connection allows you to send files back and forth between the machines: see the 'Sending and receiving files' section later for details.

Note: You can use the **'Copy'** and **'Paste'** commands on the **'Edit'** menu to copy and paste text to and from other Series 5 programs. When you paste text in the Terminal emulation screen, it is sent to the remote machine: this is often called "pasting to a host".

When you've finished the communications session, you may want to:

- Save the settings on the other machine. In HyperTerminal this is done using **'Save'** on the **'File'** menu.
- Save the communications settings on the Series 5 using **'Save settings as'** on the **'File'** menu, giving the file an appropriate name, e.g. 'Series 5 to PC'.

You will then be able to start communications between the machines again quickly, using **'Load settings'** on the **'File'** menu in Comms, and by clicking on the appropriate icon in the HyperTerminal folder.

CONNECTING VIA A MODEM LINK

Connecting the Series 5 to a remote machine via a modem link is very similar to connecting directly with a PC, except that the signals between the two machines travel via a telephone line with a modem at each end. You must ensure:

- The Baud rate you use does not exceed the maximum that the two modems are capable of handling.
- You correctly initialise the modem the Series 5 is connected to.

To start a connection:

1. Connect the Series 5 to the modem using the Docking cable and a modem adaptor cable - contact your Psion distributor for details. Connect the modem to a telephone socket.

2. Select communication settings in Comms that are appropriate to the modem's maximum speed, and identical to the remote machine's settings.
3. Start communications with the modem by typing

   ```
   AT
   ```

   and pressing Enter. The modem should return '`OK`', which will appear in the Terminal emulation screen. If it doesn't, make sure all cables are connected correctly and the communication settings are appropriate to the modem according to the manufacturer's instructions.
4. Type the modem initialisation string appropriate to the modem and press Enter. This may be '`ATZ`' or '`AT&F`', but you should check the modem manual for further details.
5. Type '`ATDT`' followed by the telephone number (including any dial-out prefix you need to add) of the remote modem you are connecting to. When the other modem answers, you will normally see a '`CONNECT`' message in the Terminal window, and can begin to log in according to the remote machine's requirements. This may require pressing the Enter key a few times to get a "prompt" (a line of text sent by the remote machine, such as '`username:`' which waits for a command or a response).

- When communicating with a remote host, you may find you receive characters too quickly to read. Use **'Pause'** on the **'Transfer'** menu to temporarily stop incoming information. Selecting **'Pause'** again resumes the transfer.

Note: If you cannot use any kind of handshaking, you may begin to lose data if the Series 5 display is paused for some time while the other machine is sending information. To avoid this, try setting a lower Baud rate that allows you to read the incoming information more easily.

CONNECTING VIA INFRARED

You can communicate with another Series 5, an infrared-enabled PC or an infrared-enabled modem using Comms and the Series 5's infrared port. To do this:

- Select **'Communication settings'** on the **'Tools'** menu and set the comms port to 'Infrared'.
- Make sure the other machine is ready to communicate with the Series 5 via infrared, then continue with your connection as you would using the Docking cable.

Since normal infrared file transfer between two Series 5 machines does not allow you to send and receive multiple files, you may want to connect directly to another Series 5 using the above method. You can then use the YModem (batch) protocol in Comms to send and receive more than one file at a time.

STOPPING ANY TYPE OF CONNECTION

- Use **'Hangup'** on the **'Transfer'** menu to close the current connection. If you are connecting via a modem link, the modem will drop the telephone line.

Comms

SENDING AND RECEIVING FILES

Once you have established communications with another machine, you can use Comms to send and receive files. The other machine must be able to use at least one of the following protocols:

- ASCII
- XModem
- YModem (batch)

ASCII protocol is adequate for plain text transfers. If you can, though, use the XModem or YModem (batch) protocol to transfer files, as these include their own error-checking. The YModem (batch) protocol also allows you to send or receive more than one file at a time.

Note: When transferring files it is important to use the same protocol as the remote machine, or information won't be interpreted correctly. You can set the protocol using the **'Protocol'** lines of the **'Send file'** and **'Receive file'** dialogs.

SENDING A FILE

To send a file to a remote machine once a connection has been established:

1. Select **'Send file'** on the **'Transfer'** menu, or tap the **'Send file'** Toolbar button.
2. Select the transfer protocol in the **'Send file'** dialog. Make sure it is one the remote machine supports and is prepared to receive.
3. Use the **'Name'**, **'Folder'** and **'Disk'** lines to locate the file you want to send.
4. Press **'OK'** on the Series 5 and prepare the other machine to receive a file. If possible, perform these actions simultaneously.

SENDING MORE THAN ONE FILE

To send more than one file:

1. Create a folder specifically for the files you want to send, place all the relevant files in it and select **'Send file'** on the **'Transfer'** menu.
2. Select the YModem (batch) protocol, then select the folder containing the files you want to send in the **'Folder'** line.
3. Press the Tab key in the **'Name'** line, hold down Shift while using the down arrow key to select all the files in the folder, then press Enter. Press **'OK'** and prepare the remote machine to receive a file.

RECEIVING A FILE

To receive a file from a remote machine once a connection has been established:

1. Select **'Receive file'** on the **'Transfer'** menu, or tap the **'Receive file'** Toolbar button.

2. Select the transfer protocol in the **'Receive file'** dialog. Make sure it is the one the remote machine will use to send the file.
3. Choose a folder for the file to be saved in. If you are receiving a file using either the ASCII or XModem protocol, specify a filename. (YModem transfers take the filename from the sending machine.)
4. Press **'OK'** on the Series 5 and instruct the remote machine to send the file. If possible, perform these actions simultaneously.

- **To receive more than one file:** follow the steps above, using the YModem (batch) protocol. Tell the other machine to send a number of files using the same protocol. At the end of the file transfer you will find you have a number of files in the target folder.

Note: When receiving .OPO files and other files which normally have extensions in their filenames, you should include these extensions in the names you give each of the received files, or the Series 5 may not be able to run them correctly.

While sending or receiving a file, a dialog will keep you informed about the progress of the transfer. When the transfer has finished, Comms will display a **'File transfer succeeded'** message and return to the Terminal emulation screen.

Note: To cancel a transfer at any time, tap the **'Cancel'** button in the file transfer dialog.

If you set the Series 5 to receive a file, it will "time out" (terminate the connection when nothing has happened for a given period) if a connection is not made within a certain time. When transmitting, however, it will not time out, and will continue waiting for a connection.

RECEIVING INFORMATION WITH NO PROTOCOL

If you are receiving information from a remote machine as plain ASCII characters, e.g. when reading e-mail on a server, you can "capture" this to a file in order to have a record. To do this:

1. Select **'Capture to file'** on the **'Transfer'** menu and give the file a name.

- If you want the new information to be added to the end of an existing file, enter the name of the existing file and tick the **'Append'** box.
- If you want incoming control characters to be recorded in angled brackets, e.g. '`<13>`' for a carriage return, tick the **'Debug'** box.

2. To end the capture, deselect the **'Capture to file'** menu option.

Captures are stored as plain text files. You can view a plain text file in Word, by using the **'More|Import text file'** command on the **'File'** menu.

If you have problems transferring information with no protocol, try setting both computers to use a lower Baud rate.

USING THE FUNCTION KEY TOOLBAR

Comms has a ‘function key toolbar’ which contains a number of buttons you can “program” to carry out commands. By setting up these buttons to carry out the commands you use when making connections, you can accelerate and simplify the process.

- To show the function key toolbar, select **‘Show toolbar|Function key toolbar’** from the **‘View’** menu.

The buttons are labelled F1 to F5; each can be used on its own, or whilst holding down Shift or Ctrl, making a total of 15 possible combinations. To assign a command to a function key:

1. Select **‘Function keys’** from the **‘Tools’** menu, and enter the “command string” for the key.
2. Use the **‘State’** line to assign more command strings for function keys when they are tapped with the Shift or Control key held down. Press **‘OK’**.

The kinds of command string you can use are as follows:

- **Text:** put text within quote marks that you want to be able to send to the remote machine easily. E.g.

  “get mail”

- **Script names:** put the full path name of a script that you want to be able to run quickly. E.g.

  C:\Commsstuff\Scripts\BBS.scr

For more information on scripts, see the ‘Scripts’ section of this User Guide.

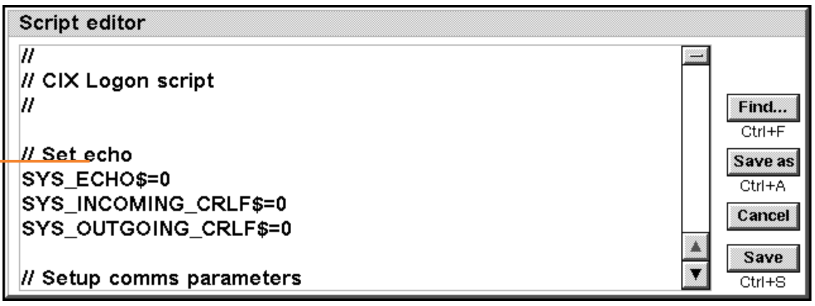
- **To use a “programmed” function key:** simply tap on the the toolbar button, or do so while holding down Shift or Ctrl as appropriate.

SCRIPTS

The Comms program contains a Script editor which allows you to create lists of instructions to automate procedures such as logging on to a remote computer. Scripts allow you to control all the aspects of a connection that you would normally handle in the Terminal emulation screen, such as communication settings and file transfers.

This section provides an introduction to creating and working with scripts, followed by a full glossary of the scripting language commands.

EXAMPLE SCRIPT



```
//
// CIX Logon script
//

// Set echo
SYS_ECHO$=0
SYS_INCOMING_CRLF$=0
SYS_OUTGOING_CRLF$=0

// Setup comms parameters
```

WORKING WITH SCRIPTS

TO CREATE A NEW SCRIPT:

1. Select **'Create new script'** from the **'Scripts'** menu. Enter a name and location for the script and press **'OK'**.

Note: Script names must have a '.scr' extension. When you save your script, Comms will add '.scr' to the end of its name.

2. Comms will move to the script editor. You can then enter your script.
3. Tap **'Save'** to return to the main Comms view.

When creating a new script for, e.g., a connection to a BBS system, it can be useful to first record the logging in process using **'Capture to file'** on the **'Transfer'** menu. This records the prompts sent to you by the remote machine when connecting, and can then be used as a starting point for creating the script.

TO EDIT AN EXISTING SCRIPT:

1. Select **'Open'** on the **'Scripts'** menu, and choose the script you want to work on. Press **'OK'**.
2. When you have finished editing the script, tap **'Save'** or use **'Save as'** to give the new version a different name. Comms will return to the Terminal emulation screen.

Note: You can adjust the font and character size of text displayed in the Script editor by pressing F whilst holding down the Ctrl and Shift keys, then selecting the appropriate options in the **'Set font'** dialog.

Comms

THE EXAMPLE SCRIPT

The Series 5 comes with an example script using a wide range of the features available in the scripting language.

- **To access the script:** Select **'Create standard files'** on the **'Tools'** menu. Select a folder for Comms to place the example script in, and press **'OK'**.

The script is called 'Cix.scr', and is designed in the first instance for connection to the CIX service, though it can easily be adapted for other dial-up services.

- **If you have a CIX account:** When you first run the script, you will be prompted for the telephone number you dial to, as well as your user name and password details. This information is stored in the 'Cixnames' names file. You are then presented with a menu offering you a number of options, e.g. 'Download mail and log off' and 'Change logon details'. If your Series 5 and modem are set up and ready to connect to another machine, you can simply select the appropriate option to start the connection.

- **If you do not have a CIX account:** You can still edit and adapt the supplied script to suit the dial-up service you use, using the **'Open script'** command on the **'Scripts'** menu.

CHECKING SCRIPTS

Comms has a built-in syntax checker that searches for errors in your scripts. It is a good idea to use this before trying to connect, as spotting any errors in advance can save a lot of time.

- **To check your script:** select **'Syntax check'** on the **'Scripts'** menu and specify the script you want to check.

If there are any errors, the Script editor will position the cursor at the appropriate point in the script and inform you of a problem.

RUNNING SCRIPTS

Once you have created a script, run it by:

- Selecting **'Start Script'** from the **'Scripts'** menu, and choosing the appropriate script in the **'Start script'** dialog.

While a script is running, you will see its name displayed in the status bar at the bottom of the Terminal emulation screen.

- **To stop a script:** select **'Stop script'** from the **'Scripts'** menu.

When a script finishes running, you are left in the Terminal emulation screen. You can run another script if you need to, simply selecting **'Start script'** again. You can also instruct one script to start another when it finishes. For information on how to do this, see the glossary entry on the **CALL** command later on.

Note: You cannot start a script running while the Comms port is inactive. Make sure **'Port active'** on the **'Transfer'** menu is ticked in the Terminal emulation screen.

AN INTRODUCTION TO THE SCRIPTING LANGUAGE

The language used in scripts is made up of commands which are very similar to ordinary English words. This section provides an introduction to some important aspects of the scripting language. The following section then guides you through all the script commands.

HOW SCRIPT COMMANDS ARE USED

A short script might look like this:

```
STATUS "Uploading test file"
UPLOAD "C:\Documents\anyfile", "Xmodem"
EXIT
```

This script uses three commands: **STATUS**, which displays an information message in the bar at the bottom of the screen; **UPLOAD**, which sends a file to the remote machine, and **EXIT**, which stops the script running, returning control to the Terminal emulation screen.

- Commands can be entered in upper or lower case, or a mixture of both.
- To combine more than one command on a line, separate them with a colon, e.g.

```
INFO "Receiving test transmission" : DOWNLOAD "C:\Temp\trashfile" , "XModem" : EXIT
```

- The total length of a line must not exceed 255 characters.

Note: In general, scripts should finish with a carriage return (i.e. ensure that there is an extra blank line at the end).

LABELS

Scripts are split up into sections marked by "labels". A label, e.g. 'mylabel:' denotes the start of a section of script. Use labels to split up scripts into manageable sections, and to enable the script to jump directly to a given area by using the **GOTO** command. E.g., in the script

```
GOTO mylabel
    STATUS "Didn't jump to mylabel"

mylabel:
    STATUS "Jumped to mylabel"
```

The first status message is never displayed, since the **GOTO** command jumps straight to 'mylabel:'

Note: To learn how to repeat the instructions given in a label a specific number of times, see the sections on the **SET** and **REPEAT...UNTIL** commands in the glossary of commands later.

END-COMMANDS

Some commands, e.g. **SETUP**, **IF** and **QUERY**, require an "end-command" to tell the machine that the instructions they refer to are finished. E.g.

```
SETUP
    BAUD=57600
    PARITY=Even
```

```
        HANDSHAKE=XONXOFF
    ENDSETUP
```

- Those commands which require ending are indicated in the glossary of script commands later on in this document.

TEXT STRINGS

Strings are sequences of characters between quotes. The **SEND** command sends a string through the serial port. E.g.

```
SEND "ATZ"
```

This sends the characters **A**, **T** and **Z**, a modem initialisation string. You can insert control characters in a string by putting them in angled brackets, e.g.

```
SEND "guest<013>"
```

sends a carriage return character after the text "**guest**". Control character codes can also be entered in hexadecimal form, prefixed with a $ sign. E.g.

```
SEND "bye<$1B>"
```

Puts the control character code 27 (the 'Esc' control) at the end of the **SEND** string.

FILENAMES AND FOLDERS

When receiving a file using the XModem protocol or ASCII, you must specify a name for the file to have on the Series 5. To do this, either

- Simply state the filename, eg.

```
DOWNLOAD "LetterIV","XModem"
```

This will place the downloaded file in the Documents folder. Or,

- Specify the full "path name", including the folder and disk, e.g.

```
DOWNLOAD "C:\Documents\Letters\LetterIV","XModem"
```

Use this method to download files directly to a memory disk, e.g.

```
DOWNLOAD "D:\Publications\Frontcover","XModem"
```

Note: Make sure the name you give to the file you are downloading is not the same as that of another file within the same directory, or you may overwrite the existing files.

When receiving files using the YModem (batch) protocol, the sending machine transmits the filename, so there is no need to specify one. Simply use a "blank text" (i.e. no characters between the quote marks) string in the command, e.g.

```
DOWNLOAD "","YModem (batch)"
```

The file will be saved in the Documents folder.

NAMES

Comms can store information you use frequently, such as passwords or telephone numbers, as “names” in a file. You can then insert a name in a **SEND** or **SENDWAIT** command which Comms will substitute with the stored information.

TO STORE INFORMATION AS A NAME:

Select **‘Set up names’** on the **‘Names’** menu in the Terminal emulation screen, and tap **‘Add’**.

Enter the name, e.g. “pass”, then enter the value, e.g. a password.

- The text represented by the name can be up to 64 characters long.
- Names can be up to sixteen characters long, and must start with a letter.

Once you have created a set of names, save them using the **‘Save names as’** command on the **‘Names’** menu. You can then use **‘Load names’** to retrieve the specific set of names that you need at any one time. In general, you should only need to use one names file, even if it contains more names than you will ever use at one time.

- To change the information for a name, use the **‘Set up names’** command, and edit the **‘Value’** line for the relevant name.

USING NAMES IN A SCRIPT

To use named information in a script, you need to instruct the machine to load the correct names file using the loadnames command, e.g.

```
LOADNAMES  “D:\Comms\Comms.nam”
```

You can then use the information in this file in either a send or sendwait command. To do this, add a **$** sign to the name. E.g., to send the information named “pass1”, use the line:

```
SEND pass1$
```

You can send more than one name at once - or mix different kinds of information - by using the **&** operator. E.g.

```
SEND username$ & pass$ & “mail”
```

- If you alter named information in the course of a connection, or create new named information using, say, the **QUERY** command, you can then use the **SAVENAMES** command to preserve the changes, e.g.

```
SAVENAMES  “C:\Comms\Names\BBSnames.nam”
```

STORING NAME INFORMATION USING THE QUERY COMMAND

You can create names while the script is running, using the **QUERY** command. E.g.

```
QUERY  “Dial-up details”
        “BBS telephone number” , no$
        “Modem initialisition string” , init$
        “Username” , name$
        “Password” , pass$
ENDQUERY
```

prompts the user with a dialog asking for information which is then stored under the names "no", "init", "name" and "pass".

RESERVED NAMES

There are three "reserved" (i.e. not free to use for any information) names, which can only be used for special kinds of information. They are:

```
sys_echo$
sys_incoming_crlf$
sys_outgoing_crlf$
```

These are essentially the scripting language equivalents of the **'Translate codes'** options on the **'Tools'** menu in the Terminal emulation screen. They toggle the "local echo" feature, and control whether or not a line feed instruction is added to each incoming and outgoing carriage return.

The line

```
sys_echo$=1
```

switches the local echo function on, which allows you to see the characters you are sending on the home screen. Setting a value of 0 switches it off.

The line

```
sys_incoming_crlf$=1
```

tells the Series 5 to start a new line each time the remote machine sends a carriage return. Setting a value of 0 switches it off.

The line

```
sys_outgoing_crlf$=1
```

tells the remote machine to start a new line each time you send a carriage return from the Series 5. Setting a value of 0 switches it off.

PASSWORD PROTECTING NAMES

You may want to protect confidential information stored as a name by giving the names file a password. To do this:

1. Select **'Password'** on the **'Names'** menu, then enter and confirm your password.
2. Select **'Save names'** on the **'Names'** menu. You will now have to enter the password to access the **'Names'** dialog.

- **To clear a password:** select **'Password'**, enter the current password, then enter a "zero-length" password in the **'Set password'** dialog.

Note: You can still run scripts that use names without entering the password. If you want to be prompted every time a script that uses password protected names runs, use the **LOADNAMES** command. E.g.

```
LOADNAMES "C:\BBSstuff\Comms.nam"
SEND init$ & phone$
```

will prompt for a password if the file being loaded is password-protected.

VARIABLES

A variable is a name which can be used to store and represent different information. E.g. **username$** could be used to represent the information “Anton”, “Sam” or “5” etc, depending on what has been stored under that name. There are two types of variables in the scripting language:

- **Non-persistent variables:** these have no **$** after them, e.g. “**phoneno**”. Information stored in this kind of variable is lost when the script stops running.
- **Persistent variables:** these are followed by a **$**. Information stored in this kind of variable can still be edited and saved once the script has stopped running, using the **‘Set up names’** and **‘Save names as’** commands on the **‘Names’** menu.

SCRIPT COMMANDS

This section provides a summary of the script commands according to type, followed by a detailed glossary of commands and their usage.

SUMMARY OF THE SCRIPT COMMANDS

PORT AND HANDSHAKING CONTROL

CONNECT
HANGUP
LINE
REPLY
RESET
SETUP

SENDING AND TESTING FOR CHARACTER STRINGS

SEND
SENDBREAK
SENDWAIT
WAIT

STRING OPERATIONS

&
ASC
COLLATE
FOLD
LEFT
LOWER
MID
RIGHT
UPPER

Comms

FILE HANDLING

APPEND
AS
CAPTURE
CATCH
CLOSE
COPY
DELETE
DOWNLOAD
EOF
EXCLUDE
EXISTS
INCLUDE
LOADNAMES
MOVE
OPEN
READ
RENAME
SAVENAMES
UPLOAD
WRITE

USER INFORMATION & INTERACTION

ALERT
BEEP
CLS
INFO
MENU
QUERY
QUERYOK
STATUS

PROGRAM CONTROL

BREAK
CALL
CHAIN
CONTINUE
ELSE
ELSEIF
EXIT
FORGET
GOTO
IF
ON
THEN
WHILE
DO
REPEAT
UNTIL
SET

Comms

LOGICAL OPERATORS

AND
NOT
OR

EXPRESSION OPERATORS

= - / * =
<>
<
>
<=
>=
%

OTHER

//
DRAIN
VERSION

GLOSSARY OF COMMANDS

This section contains an alphabetical list of all the commands available when creating scripts. Examples are given for each command. You may also find it useful to look at the example script supplied with the machine.

Note: Some commands, like **SETUP** and **MENU**, must be ended by a corresponding command, e.g. **ENDSETUP** or **ENDMENU**. It is indicated below where this is necessary.

HOW GLOSSARY ENTRIES ARE ARRANGED

Glossary entries on script commands consist of three parts:

- **The name of the command itself.** This is on the left side of the page. Related commands and other commands explained in the same glossary entry are listed underneath.
- **The command syntax**. This appears to the right of the command name. The syntax indicates the other types of information that are needed for the command to operate, and how the components should be arranged.
- **The command definition.** This is found below the command name and syntax. The definition explains exactly what the command does, and gives examples of its uses.

SYNTAX ABBREVIATIONS

The explanations of command syntax use a number of terms to denote different types of information:

- **<value>:** indicates a number.
- **<string>:** indicates a text string within quotation marks.

- **<string exp>:** indicates either text within quotation marks or a variable which contains text (e.g. **user$** if the information it represents is, say, “Adam”).
- **<variable>:** indicates either a persistent or non-persistent variable, e.g. **mailpass$** or **modeminit**.
- **<filename>:** indicates a filename within quotation marks. It is usually better to include the full path in filenames, e.g **“C:\Documents\Logs\Log1”** rather than **“Log1”**.
- **<handle>:** indicates a numeric label you have assigned to a file you are opening. You can open up to 9 files, using the handles 1-9.
- **<label>:** indicates a script label.
- **<command>:** indicates a scripting language command.
- **<exp>**: indicates an expression, e.g. a condition in an **IF…THEN** command such as ‘**day$=“Tuesday”**’.

Note: If the syntax explanation just contains the command on its own, e.g. **DRAIN**, then the command does not require any additional information when used in a script.

- Where the command syntax contains square brackets, the elements within brackets are optional.

COMMANDS

&

see **SEND**

//

//

Precedes a comment you have included to explain what part of a script does. Everything from the ‘//’ to the end of the line is ignored. You can also put a comment after another command, e.g.

```
SEND “Fred”,“XModem” // Sends password
```

(There is no need to insert a colon.)

ALERT

ALERT <string exp>[,<string exp>]

Prompts the user with information (indicated by the text string) and waits for acknowledgement before continuing. E.g.

```
ALERT “No carrier - try again later.”
```

You can add a second line to the alert dialog using an optional second string of text. E.g.

```
ALERT “No carrier.”,“Check modem is connected correctly and try again.”
```

And
Not
Or

AND
NOT
OR

Use the **AND**, **NOT** and **OR** logical operators to link conditions in an expression. E.g.

```
IF c>1 AND c<10 THEN
        INFO “Value is between 1 and 10”
ELSE
        INFO “Value is not between 1 and 10”
ENDIF
```

only carries out the first **INFO** command if *both* conditions are met. The following section of script

```
IF usern$=“Jerry” OR usern$=“Sam” THEN
        pass$=“disk14”
ELSE
        GOTO getpass
ENDIF
SEND pass$
```

defines **pass$** and sends **pass$** if *either* of the conditions are met. You can use **NOT** to specify conditions which must *not* be met for a given action to be carried out.

```
IF username$=“” AND NOT superuser$=“Anton99” THEN
        GOTO getuserinfo
ELSEIF superuser$=“Anton99” THEN
        GOTO superlog
ELSE
        GOTO getpass
ENDIF
```

Here the script will jump to ‘**getuserinfo**’ only the variable ‘**username$**’ contains no information, and the variable ‘**superuser**’ does *not* contain the information ‘**Anton99**’.

As

see **Open**

Asc

ASC (<string exp>)

Returns the **ASCII** code of a character. If a whole string is entered between the brackets, **ASC** returns the code for the first character unless you specify a particular character using **LEFT**, **MID** or **RIGHT**. E.g.

```
INFO ASC (“H”)
```

returns the code 72.

Comms

Beep

BEEP <value>,<value>

Activates the Series 5's buzzer. The first value is the length of the note in 1/32s of a second, and the second indicates the pitch. E.g.

```
BEEP 16,300
ALERT "Connection dropped"
```

Break

BREAK

Breaks out of a loop.

Call

CALL <filename>

Stops running the current script and transfers control to a new one, without clearing any variables. E.g.

```
CALL "C:\Documents\Scripts\script2.scr"
```

starts 'script2.scr' running. When the called script ends, the Terminal emulation screen is shown. The program does not return to the calling script.

Capture
Capture off
Append
Debug

CAPTURE <filename>,[APPEND[,DEBUG]]
CAPTURE OFF

Begins saving received characters to a specified file. Use **CAPTURE OFF** to stop the capture and save the file. E.g.

```
CAPTURE "C:\Documents\Logs\log15"
...
CAPTURE OFF
```

To add more captured information to a file that exists already, use the **APPEND** modifier. E.g.

```
CAPTURE "C:\Documents\Logs\log", APPEND
```

This simply adds the newly captured information onto the end of the old. You may find **CAPTURE** useful when reading your e-mail messages on a remote system. E.g.

```
CAPTURE "C:\Email\messages"
SENDWAIT 600 "readmail" , "ready:" GOTO fail5min
CAPTURE OFF
```

Sends "**readmail**" to the remote system - a command to 'read my messages' - and captures the resulting text. In this system, the remote machine signals the end of the mail by a return to the "**ready:**" prompt. If this happens within 5 minutes, the capture is terminated, if not, the script jumps to the '**fail5min**' label.

Note: To store control characters in the captured file, e.g. <13> or <8>, use the **DEBUG** modifier. E.g.

```
CAPTURE "C:\Scripting\log" , DEBUG
```

You can combine the two modifiers together by separating them with a comma. E.g.

```
CAPTURE "C:\Captures\log2" , APPEND , DEBUG
```

CATCH

CATCH [INCLUDE/EXCLUDE] ,<string>, <variable>, <timeout>, <maxlength> GOTO <label>

Stores incoming information to the specified variable. E.g.

```
CATCH INCLUDE, "OK" , R$ , 60 , 20 GOTO anerror
```

This captures incoming data to the variable **R$**. The capturing will stop after a time of **<timeout>** (in this case 60 half-seconds), when the string "OK" is received, or when twenty characters have been received. The script jumps to the label (i.e. '**anerror**') if the given string is not received before either the **<timeout>** value or the maximum number of characters (**<maxlength>**) is reached. Use **INCLUDE** if you want **<string>** to be included as part of the variable, or **EXCLUDE** if you do not.

CHAIN

CHAIN <filename>

Starts a new script, clearing all variables first. E.g

```
CHAIN "C:\Documents\Scripts\script3.scr"
```

starts 'script3.scr' running, whilst clearing all the variables set up in any previous scripts.

CLOSE

see **OPEN**

CLS

CLS

Clears the terminal screen.

COLLATE

COLLATE (<string exp>)

Collates a string, by removing the differences between the characters that are deemed unimportant for the purposes of sorting them.

Comms

Connect

CONNECT <value> GOTO <label>

Waits for the current handshaking method to allow transmission. The period to wait is specified in half-seconds. E.g.

```
CONNECT 20 GOTO again
SEND “Hello”
```

waits 10 seconds for the connection to be established. If it happens within this time, the script jumps to the next command (“**SEND**”), otherwise it moves to the ‘**again**’ label.

Continue

CONTINUE

Makes the script jump to the start of a **REPEAT…UNTIL** or **WHILE…DO…ENDWHILE** loop.

Copy

COPY <filename>,<filename>

Copies a file, from the location specified in the first string expression to the location specified in the second. E.g.

```
COPY “C:\Logs\Logfile3”,“D:\Documents\logfile3”
```

will create a copy of the file ‘Logfile3’ in the specified directory on the D: drive.

Delete

DELETE <filename>

Deletes a specified file. E.g.

```
DELETE “C:\Logs\Logfile3”
```

Dir

DIR (<string exp>)

Returns a directory listing of files. To list all the files in a directory, use the following sequence of commands:

```
X$=DIR(“C:\Documents\scripts\*.*”)
WHILE X$<>“” DO
        INFO X$
        X$=DIR
ENDWHILE
```

You can list specific files, say script files, by limiting the string expression. E.g.

```
DIR (“C:\Documents\*.scr”)
```

which will only list files with the filename extension ‘.scr’.

Do

see **While**

Comms

Download

DOWNLOAD <filename>,<string>

Prepares the Series 5 to receive a file from the remote machine, using a protocol indicated by the second string expression. E.g.

```
DOWNLOAD "D:\Documents\letter1" , "XModem"
```

The protocol is indicated by one of the following labels: ASCII; XModem; YModem (batch). Both XModem and ASCII protocols require you to specify a filename in the first string expression, as above. When receiving files using the YModem (batch) protocol, the filename is sent by the other machine, so a "blank text" string ought to be entered. E.g.

```
DOWNLOAD "","Ymodem (batch)"
```

Drain

DRAIN

Empties the "receive buffer" (the part of the memory where received characters are stored before being displayed). Any received characters which have yet to be processed will be discarded.

EOF

see **OPEN**

Exists

EXISTS <filename>

Returns a value of 1 or 0, depending on whether or not a file exists. If the file exists, the script carries out the commands specified by an **IF** command. E.g.

```
IF EXISTS "C:\Files\Log"=1 THEN
        GOTO gotlog
ENDIF
GOTO nolog
```

checks to see if there is a file called 'Log' in the 'Files' folder. If there is (i.e. if **EXISTS** returns a value of 1), the script jumps to the '**gotlog**' label. If there is no such file (and **EXISTS** returns 0), the script jumps to the '**nolog**' label. **EXISTS** can also be used to assign a value to a variable. E.g.

```
SEND EXISTS "C:\Files\Log"
```

will send 0 or 1 depending on whether or not the file exists, while

```
gotlogfile$=EXISTS "C:\Files\Log"
```

sets the value of the string according to the value returned by **EXISTS**.

EXIT

EXIT

Stops running the script and returns to the Terminal emulation screen. E.g.

```
endsession:
        INFO "End of session. Now hanging up."
        HANGUP
        EXIT
```

FALSE

see **TRUE**

FOLD

FOLD <string exp>

"Folds" a string: i.e. converts it to upper case, and removes any accents. E.g.

```
SEND FOLD Username$
```

FOR

see **OPEN**

FORGET

FORGET

Clears all variables.

GOTO

GOTO <label>

Jumps to the command following a specified label. E.g.

```
GOTO prompt
...
prompt:
        WAIT 60
        "Hello" GOTO gothello
        ...
        ENDWAIT
```

The **GOTO** command makes the script jump straight to the **WAIT...ENDWAIT** command. The specified label can be anywhere in the script.

HANGUP

HANGUP

Drops the DTR line for three seconds. If you are currently using a telephone line connection, your modem will hang up and drop the DCD line to the Psion. This command has the same effect as **'Hangup'** on the **'Transfer'** menu.

IF / THEN / ELSEIF / ELSE / ENDIF

IF <exp>THEN<command>[[ELSE]<command>[ELSEIF<exp>THEN<command>]] ENDIF

Allows you to specify a choice of instructions according to the conditions. **IF** requires both a **THEN** command, and an **ENDIF** command. E.g.

```
IF Username$=“” THEN
        GOTO getuserinfo
ELSEIF Username$=“Bob” THEN
        GOTO boblogon
ELSE GOTO getpass
ENDIF
```

checks to see if there is any information stored in the ‘**Username**’ name. If there is no such information, the script jumps to the ‘**getuserinfo**’ label. If the name contains the information ‘**Bob**’, the script jumps to the ‘**boblogon**’ label. If the name contains something other than ‘**Bob**’, it jumps to the ‘**getpass**’ label. You can link more than one condition to an **IF** command by using the **OR** and **AND** logical operators. E.g.

```
IF Username$=“” OR Password$=“” THEN
        GOTO getuserinfo
```

jumps to the ‘**getuserinfo**’ label if *either* of the names has no information associated with it. On the other hand,

```
IF Username$=“” AND Address$=“” THEN
        GOTO getuserinfo
```

jumps to the ‘**getuserinfo**’ label only if *both* of the names have no associated information.

INFO

INFO <string exp>

Displays a message in the terminal screen, e.g. about the action currently being performed. E.g.

```
INFO “Initialising modem and dialling out...”
```

You can use **INFO** messages to keep you informed about the progress of your connection.

INPUT

see **OPEN**

LEFT / RIGHT

LEFT (<string exp>,<value>)

RIGHT (<string exp>,<value>)

Extracts a specified number of characters from the furthest left or right part of string. E.g.

```
SEND LEFT (logname$,3)
```

will send “**sec**” if the information stored as **logname$** is “**secondlog**”. Similarly,

```
SEND RIGHT (logname$,3)
```

would send “**log**”.

Len

LEN <string exp>

Returns the length of a string. E.g.

```
namelen=LEN (name$)
```

will assign a value of 5 to ‘**namelen**’ if the string “**Anton**” is stored as **name$**.

LoadNames
SaveNames

LOADNAMES <filename>

SAVENAMES <filename>

Loads a given names file, so that you can use its information in the current script. E.g.

```
LOADNAMES “C:\Documents\Comms\BBSnames”
```

SAVENAMES saves all the current names information in a specified file. E.g.

```
SAVENAMES “C:\Documents\Names\Newnames”
```

Lower

see **Upper**

Menu
EndMenu

```
MENU <string exp>
    <string exp> GOTO <label>
    <string exp> GOTO <label>
    ...
ENDMENU
```

Offers a menu of options to the user. Each option jumps to a particular label when it is chosen. E.g.

```
MENU “Choose action”
     “Connect to server” GOTO logon
     “Upload file” GOTO sendfile
     “Download file” GOTO downl
     “Exit” EXIT
ENDMENU
```

creates a menu with the heading “Choose action” from which the user can select one of four actions (e.g. “Upload file” etc). The script then carries out the command next to the menu item, e.g. jumping to “sendfile” for the “Upload file” menu item. Menu items can only be followed by **GOTO** or **EXIT** commands. **MENU** requires a matching **ENDMENU** command.

Mid

MID (<string exp>,<value>[,<value>])

Extracts part of a string or variable, beginning a specified number of characters (indicated by the first value) from the left. E.g.

```
logfilename$="C:\files\log1"
INFO "..." & MID (logfilename$,9)
```

will display "...log1" in the terminal screen. The optional second value allows you to specify the number of letters to be extracted. E.g.

```
SEND MID ("password15",4,4)
```

will just send the characters '**word**'.

Move

MOVE <filename>,<string exp>

Moves a file, from the location specified in the first string expression to the location specified in the second. E.g.

```
MOVE  "C:\Logs\Capture","C:\Email\"
```

Not

see **And**

On error goto
On error off

ON ERROR GOTO <label>
ON ERROR OFF

Makes the script jump to the specified label if an error occurs. The line

```
ON ERROR OFF
```

returns Comms to normal error handling.

Open
Input
Output
As
For
Read
Write
Eof
Close

```
OPEN <filename> FOR [INPUT/OUTPUT/APPEND] AS <handle>
    READ <length>,<variable>,<handle>
    WRITE <handle>,<string>
    EOF <handle>
CLOSE <handle>
```

Opens files and enables reading from ("input") and writing to ("output") them. When you open a file, you assign it a "handle" (a number from 1 to 9), which you then use to refer to that file. You can have up to 9 files open at a time. E.g., to write a text string to a file:

```
OPEN "C:\Files\Connectlog" FOR OUTPUT AS 1
     WRITE 1,"Connect failed"
CLOSE 1
```

opens the file “Connectlog” to write to it, then writes “Connect failed” and closes it again. You can use **OPEN** to read information from one file and write it to another. When reading from a file, use **EOF** to look for the end of the file, so that Comms knows when to stop. E.g.

```
OPEN “C:\sourcefile” FOR INPUT AS 1
OPEN “C:\destination” FOR OUTPUT AS 2
WHILE NOT EOF(1) DO
        READ 16,a$,1
        WRITE 2,a$
ENDWHILE
CLOSE 1
CLOSE 2
```

This script opens the two files, then instructs Comms to read 16 characters into **a$** from the first file and write the contents of **a$** to the second file. These instructions are held within a **WHILE...DO...ENDWHILE** loop, so they are repeated until the end of the first file is reached.

OR

see **AND**

OUTPUT

see **OPEN**

QUERY
QUERYOK
ENDQUERY

```
QUERY <string exp>
      <string exp>,<variable>
      <string exp>,<variable>
ENDQUERY
```

Prompts the user for information which is then stored as a name. E.g.

```
QUERY “Enter details”
        “Username:” , myname$
        “City” , city$
ENDQUERY
```

creates a dialog where the user can enter information such as “Username” etc. This information is then stored with the name that follows, e.g. ‘**myname**’. Up to six items can be prompted for in a **QUERY** dialog. The string which follows **QUERY** is the title of the dialog. **QUERY** requires a matching **ENDQUERY** command. **QUERYOK** allows the script to act differently depending on whether the **QUERY** dialog was exited by tapping **‘OK’**, or by pressing Esc. E.g.

```
QUERY “Enter details”
        “Username:” , myname$
        “City” , city$
        “Password” , pass$
ENDQUERY
IF QUERY OK THEN
```

```
        GOTO logon
ELSE
        GOTO cancelled
ENDIF
```

jumps to the ‘**logon**’ label if **‘OK’** is pressed, but the ‘**cancelled**’ label if the **QUERY** dialog is exited using Esc.

READ

see **OPEN**

RENAME

RENAME <filename>,<filename>

Renames the file indicated in the first string expression to the name specified in the second. E.g.

```
RENAME "C:\Documents\Names","Names2"
```

REPEAT UNTIL

```
REPEAT
    <command>
UNTIL <expression>
```

Repeats a command until a certain condition is met. E.g.

```
SET a=5
REPEAT
        SEND "<013>"
        a=a-1
UNTIL a=0
```

Sends carriage returns and deducts 1 from the value of **a** until the value of **a** reaches 0.

Note: The important difference between **REPEAT…UNTIL** loops and **WHILE…DO…ENDWHILE** loops is that **REPEAT…UNTIL** always executes the specified action at least once, and ceases to do so when a condition is met. By contrast, **WHILE…DO…ENDWHILE** checks for a condition *before* ever carrying out an action, so if the condition is not met, the action will not be executed at all.

RESET

RESET

Returns all communications settings back to their standard values. You may want to use this before a **SETUP** command, so that you can be sure of the complete configuration. E.g.

```
RESET
SETUP
        Baud=57600
        Parity=Even
ENDSETUP
```

Comms

RIGHT

see **LEFT**

SAVENAMES

see **LOADNAMES**

SEND
&

SEND <string exp> [;] [& <string exp>]

Sends data to a remote machine, usually for controlling a modem or for conveying information to a remote system. E.g.

```
SEND “atz”
```

sends a modem initialisation string. You can add the character code for the Enter key to the **SEND** command by putting a semi-colon after the string, e.g.

```
SEND “mypassword”;
```

To include control characters, such as carriage returns or line feeds, in a **SEND** string, use angled brackets. E.g.

```
SEND “mail<13><10>”
```

sends a carriage return (control character number 13) and then a line feed (control character number 10) after ‘**mail**’. Control character codes can also be entered in hexa-decimal form, prefixed with a **$** sign. E.g.

```
SEND “bye<$1B>”
```

puts the control character code 27 (the ‘Esc’ control) at the end of the **SEND** string. You can send more than one string at a time by using the **&** operator. E.g.

```
SEND “john” & “password29”
```

SENDBREAK

SENDBREAK

Sends a “break” (i.e. transmits nothing for a short space of time) to the remote machine. The effect of this depends on the type of remote machine and the software it uses.

SENDWAIT

SENDWAIT <value> <string exp>,<string exp> GOTO <label>

Combines a **SEND** command with a **WAIT** command. E.g.

```
SENDWAIT 60 “Anton” , “Password:” GOTO nopass
GOTO pass
```

does exactly the same as

```
SEND “Anton”
WAIT 60
     “Password:” GOTO pass
```

```
GOTO nopass
...
```

I.e. the Series 5 sends “**Anton**”, then waits for 30 seconds (60 half-seconds) for the remote machine to send “Password:”. If “Password:” is received, the script jumps to the ‘**pass**’ label. If “Password:” is not received, the script moves on to the next command (‘**GOTO nopass**’). The script only follows the **GOTO** command after “Password:” if “Password:” *isn’t* received.

SET

SET <variable>=<value>

Set assigns a variable or name a given value. E.g.

```
SET me$=“Albert”
SET c1=5
```

Assigns the information “**Albert**” to the name ‘**me**’ It also gives the variable **c1** a value of 5. You can then decrement this value to control the number of times an action is performed. E.g.

```
setcount:
    SET c1=5 sendcr:
    SEND “<13>”
    c1=c1-1
    IF c1=0 THEN
        GOTO next
    ELSE
        GOTO sendcr
    ENDIF
next:
...
```

first gives **c1** a value of 5. The script then sends a carriage return character (“**<13>**”) and deducts 1 from the value of **c1**. If **c1** still hasn’t reached 0, it repeats this step. Once it has sent 5 carriage returns, it will move on to the ‘**next:**’ label.

SETUP
BAUD
PORT
DATA
FAIL
HANDSHAKE
PARITY
STOP
TIMEOUT
ENDSETUP

```
SETUP
   BAUD [rate]
   PORT [COMM::0][IRCOMM::0]
   DATA [5][6][7][8]
   FAIL [DSR][DCD][PARITY]
   HANDSHAKE [DCD][RTSCTS][DSR][XONXOFF][NONE]
   PARITY [NONE][ODD][EVEN]
   STOP [1][2]
   TIMEOUT [timeout]
ENDSETUP
```

Sets any or all of the communications parameters. All of the above parameters can be set, to any of the options within square brackets. You can separate each of the parameters either with a colon, or by putting them on a new line. E.g.

```
SETUP
        BAUD=19200 : STOP 1
        DATA=8
        PORT="COMM::0"
ENDSETUP
```

Note: For both **HANDSHAKE** and **FAIL**, you can set more than one of the choices by separating them with a comma. E.g.

```
HANDSHAKE=XONXOFF,DCD
FAIL=DCD,DSR
```

Parameters you do not specifically set with **SETUP** remain unchanged. **SETUP** requires a matching **ENDSETUP** command.

STATUS

STATUS <string exp>

Displays an information message in the status bar at the bottom of the Terminal emulation screen. Use **STATUS** messages to keep you informed about the progress and state of your connection, e.g.

```
STATUS "Connected to BBS."
```

TRUE
FALSE

TRUE

FALSE

Return 1 if a specified condition is true, and 0 if it is false.

```
IF user$="bob" THEN
        f$=TRUE
ELSE
        f$=FALSE
ENDIF
SEND f$
```

Will send **1** if the information stored as **user$** is "**bob**", and **0** if it is not.

UNTIL

see **REPEAT**

UPLOAD

UPLOAD <filename>,<string>

Prepares the Series 5 to send a file to a remote machine, using a protocol indicated by the second string. E.g.

```
UPLOAD "D:\Documents\letter1" , "XModem"
```

The protocol is indicated by one of the following labels: ASCII; XModem; YModem (batch).

Upper
Lower

UPPER <string exp>

LOWER <string exp>

Convert text to upper or lower case. E.g.

```
SEND UPPER username$
```

will send "**CLIVE**" even if the information stored as "**username$**" is "**Clive**". Similarly,

```
INFO LOWER prompt$
```

will display the information in lower case. You can combine **UPPER** with other text control commands such as **MID**, **LEFT**, **RIGHT** etc. E.g.

```
STATUS UPPER LEFT (L$,1) & LOWER MID (mid(L$,1))
```

will display **L$** with a capital letter at the start and the rest in lower case.

Version

VERSION

Returns the version number of Comms.

Wait
Endwait

```
WAIT <value>
    <string exp> GOTO <label>
    <string exp> GOTO <label>
    ...
ENDWAIT
```

Pauses the script for a given number of half-seconds, and waits for one of several strings. E.g.

```
WAIT 10
     "No carrier" GOTO tryagain
     "Hangup" GOTO closeconn
ENDWAIT
SEND "ATDT91011992220"
```

pauses for 5 seconds before proceeding with the next command in the script ('**SEND**'). If, during that time, the Series 5 receives "**No carrier**" from the remote machine, it will carry out the '**GOTO tryagain**' command. Similarly, if the Series 5 receives "**Hangup**" during the 5 seconds, the script will jump to the '**closeconn**' label. **WAIT** requires a matching **ENDWAIT** command.

While
Do
Endwhile

```
WHILE <exp> DO
    ...
ENDWHILE
```

Carries out a given command while a certain condition is met. E.g.

```
WHILE R$<>"OK" DO
        SEND "AT"
ENDWHILE
```

sends the "**AT**" string as long as **R$** is not equal to "**OK**".

Note: The important difference between **REPEAT…UNTIL** loops and **WHILE…DO…ENDWHILE** loops is that **REPEAT…UNTIL** always executes the specified action at least once, and ceases to do so when a condition is met. By contrast, **WHILE…DO…ENDWHILE** checks for a condition *before* ever carrying out an action, so if the condition is not met, the action will not be executed at all.

WRITE

see **OPEN**

INDEX

Comms

Comms