HEARSAY

STATE OF THE ART

COMMUNICATIONS SOFTWARE

RISC developments

HEARSAY

STATE OF THE ART

COMMUNICATIONS SOFTWARE

RISC developments

HEARSAY

User Guide

RISC developments Ltd

While every care is taken, the publisher cannot be held responsible for any errors in this product, or for the loss of data or consequential effects from the use of this package.

If you discover a problem with this product, please write to the Software Manager at the address given below describing the problem as briefly as possible, but including all relevant information e.g. version of software you are using, system configuration, hardware add-ons, version of operating system etc. Do not return your master discs or packaging until requested to do so.

Hearsay II was written by D. Pilling.

SECOND EDITION 1993

Published by RISC Developments Ltd.
117 Hatfield Road, St. Albans, Herts. AL1 4JS England
Telephone: (0727) 840303 Fax: (0727) 860263

Contents

1 Introduction

Hearsay II is the latest development of the definitive communications package for Acorn RISC based systems. The original version of the software was successful because it provided a range of powerful features that were both easy-to-learn and easy-to-use. This philosophy has been adhered to in the design of Hearsay II, which provides even more advanced features with an improved user interface. Hearsay II is multitasking and fully RISC OS compliant.

The high quality terminal emulations provided in the original version have been improved considerably, and its strong file transfer facilities have been strengthened with the addition of the Zmodem protocol.

The main features of Hearsay II are:

- Fully RISC OS compliant.
- Multitasking, including background data transfer.
- VT320, VT102, VT52, ANSI and Teletype scrolling text terminals.
- Viewdata and Minitel (CEPT 2) terminals. Viewdata frame editor, telesoftware downloading and frame tagging.
- Advanced Tektronix 4105 colour graphics terminal.
- Scalable terminal windows in all screen modes.
- Xmodem, Xmodem 1K, Ymodem, Zmodem, Kermit, SEAlink and ASCII file transfer protocols, including batch transfers.
- Campus 2000 terminal.
- Comprehensive script language based on a subset of C.
- Macro processor, and fully definable keyboard.
- Support for RISC OS printer drivers.
- Number directory with auto-logon and password protection.
- Vasscom and MNP link level error correction.
- Modem drivers for most popular modems. Additional drivers may be written in the script language and added to the system.
- User menu for customised applications.
- Call logging.
- GIF file previewer

How to use this guide

All users should read the chapter *Getting started*, which describes how to run Hearsay, configure it, and get online. If you are new to communications, you should consult the *Glossary* at the end of this user guide if you encounter any unfamiliar terms. Once you are online, you should consult the chapters *Text terminals*, *Viewdata terminal* and *Tektronix terminal*, which describe the operation of the terminals in detail. Chapters 3 to 7 describe all the other menu options in Hearsay, and may be read as required. The appendices and the chapters *Macros* and *CScript*, which describe the macro facility and script language, are intended for more advanced users.

This user guide assumes that you are familiar with the RISC OS desktop and filing systems. If you are not, please read the relevant sections of your computer user guides.

For clarity, Hearsay II is referred to as simply Hearsay throughout this user guide.

Typographical conventions

Bold type is used for emphasis and for menu and dialogue box options:

> Click on **Receive file** to download the file.
> **Always** ensure you have set the correct baud rate.

Italic type is used to indicate a term appears in the Glossary, or a cross reference to another part of the user guide or to other publications:

> A *modem* is used to transfer data across the telephone network.
> Refer to the chapter *File transfer* for more details.

Fixed-width Courier type is used for filenames, script commands, and for text that the computer displays:

> Double-click on `!Hearsay` to load the application.
> The function `sputc()` writes a character to the serial port.
> The message `Unimplemented server command` is displayed.

Latest information

Hearsay is supplied with a set of release notes which give details of any changes made since this user guide was printed. Please read the release notes carefully before proceeding. In addition, the file `ReadMe` on the Program disc lists the latest additions and changes. Double-click on this file to read it.

2 Getting started

The Hearsay package contains the items listed below. If any parts are missing please contact your supplier or RISC Developments.

- This user guide.
- Hearsay Program disc, containing:

`!Hearsay`	The Hearsay program and its resources.
`!System`	Contains modules required by Hearsay.
`ReadMe`	A file listing the latest changes and additions.

- Hearsay Extras disc, containing:

`!ANSIpal`	True ANSI palette.
`!ConMan`	Converts files to Hearsay II (see release notes).
`!Preview`	GIF file viewer (see release notes).
`!SparkPlug`	File de-archiving utility (see *Appendix F*).
`!SysMerge`	Merges new `!System` directory with your old one.
`Arc`	Public domain file archive utility (see *Appendix F*).
`PDrivers`	Printer drivers for use with RISC OS 2.
`Scripts`	Example script files (see `Scripts.ReadMe`).
`SpoolFiles`	Example sessions (see `SpoolFiles.ReadMe`).
`Tek`	Outline font Tek, for use in the Tektronix terminal.
`Utilities`	General utilities (see `Utilities.ReadMe`).

- A set of release notes that give information about any changes made to Hearsay since this user guide was printed.

Your purchase of Hearsay entitles you to make one backup copy for your own use. If you wish to make several backups for use by several users in your establishment, you should obtain a suitable site licence from RISC Developments.

It is illegal to supply copies of Hearsay to a third party.

Hardware requirements

To use Hearsay, the minimum requirements are:

- An Acorn 32-bit RISC system with at least 2Mb of RAM and fitted with the RISC OS operating system. A3000 users must ensure that the serial upgrade has been fitted.
- A standard resolution monochrome or colour monitor. Hearsay may also be used with high-resolution monochrome or colour multi-scan monitors, and contains high definition character sets for use in these modes. Hearsay will also work with very high-resolution monochrome monitors which operate in screen modes such as mode 23.
- A single floppy disc drive. Details of how to install Hearsay onto floppy disc and hard disc are given later in this chapter.
- A *modem* suitable for Acorn systems, and a connecting lead. Further details about connecting modems are given later in this chapter and typical wiring diagrams are given in the release notes together with details of the modems currently supported.
- Optionally you may require a printer, or access to a printer so that you can print the data you receive. Hearsay can print directly to the printer connected to your printer port, or via the standard RISC OS printer drivers. If you have RISC OS 3 you should use the printer drivers supplied with it. If you have RISC OS 2, use the version 2 printer drivers supplied on the Hearsay Extras disc or those available from other suppliers. Hearsay cannot print using the original version 1 printer drivers.

Running from floppy disc

Before using Hearsay, make a backup copy of the master discs onto newly formatted discs, label them correctly, and put the originals in a safe place.

The instructions below, describe how to run Hearsay from floppy disc.

1. If you intend to use the *Tektronix* terminal you should copy the font `Tek` on the Extras disc into your usual `!Fonts` directory and open a directory display on `!Fonts` before running Hearsay.

   If you are using RISC OS 2, you must ensure that the `!Fonts` directory contains a copy of the **outline** font manager i.e. version 2.44 or later. If in doubt, please contact RISC Developments for advice.

2. Insert the Program disc into drive 0 and click on the floppy drive 0 icon on the icon bar. A directory display will open showing the files on that disc. Double-click on the `!Hearsay` to load the application. It will take a few seconds to load, after which the Hearsay icon will appear on the icon bar.

   If there is insufficient memory, an error message will be displayed. If this happens, remove any other applications that may be running before trying to load Hearsay again.

3. If you intend to print anything in the current session, you may need to load a printer driver. Hearsay can print directly to your printer without the need for a printer driver, making it quicker and saving memory. However, if your printer is not Epson or IBM compatible, you will need to load a suitable RISC OS printer driver. If you have RISC OS 3, load a suitable printer driver according to the RISC OS 3 user guide.

   If you have RISC OS 2, remove the Program disc from drive 0 and insert the Extras disc. Click on the floppy drive 0 icon on the icon bar, and double click on the `PDrivers` directory. Now double-click on the required printer driver. You may use other printer drivers not supplied with Hearsay, but they must be release 2 versions supporting 'fancy' text printing.

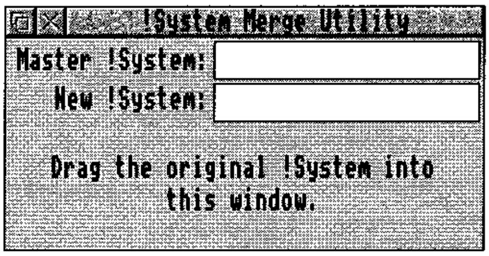
4. If you want to save anything to disc during the session, you should insert a suitably formatted disc into drive 0.

Installation on hard disc

The following instructions give the recommended procedure for installing on a hard disc system.

1. Create a directory called `Hearsay` (or use a name of your choice) in the hard disc `$` directory, and copy `!Hearsay` and `ReadMe` from the Program disc into it.
2. If you are using RISC OS 2 and do not have the latest printer drivers (version 2.44 or later) installed on your hard disc, copy the directory `PDrivers` from the Extras disc into your hard disc `$` directory.
3. If you intend to use the *Tektronix* terminal you should copy the font `Tek` on the Extras disc into your usual `!Fonts` directory.

   If you are using RISC OS 2, you must ensure that the `!Fonts` directory contains a copy of the **outline** font manager i.e. version 2.44 or later. If in doubt, please contact RISC Developments for advice.
3. Copy everything except `PDrivers` and `Tek` from the Extras disc in the `Hearsay` directory created in step 1 above.
4. Hearsay requires a number of modules to be loaded before it will run. These modules are supplied in the `!System` directory, which should be merged with your usual `!System` directory on the hard disc. The utility SysMerge supplied on the Extras disc should be used for this purpose.



!System Merge Utility

Master !System:

New !System:

Drag the original !System into this window.

Double-click on `!SysMerge` and drag your master `!System` directory into the dialogue box; its pathname will be displayed in the upper box. Now drag `!System` from the Hearsay Program disc into the dialogue box; its pathname will be displayed in the lower box. If there are no problems the two system directories will now be consolidated and the message `!System Updated` displayed. You can quit this application by clicking on the close icon.

Running from hard disc

Once installed, Hearsay may be run as follows:

1. Click on the hard drive icon on the icon bar to open a directory display showing all the files in the hard disc root directory. Open the directory containing the the Hearsay application, and double-click on `!Hearsay`. After a few seconds the Hearsay icon will appear on the icon bar.

   If there is insufficient memory, an error message will be displayed. If this happens, remove any other applications that may be running before trying to load Hearsay again.

2. If you intend to print anything in the current session, you may need to load a printer driver. Hearsay can print directly to your printer without the need for a printer driver, making it quicker and saving memory. However, if your printer is not Epson or IBM compatible, you will need to load a suitable RISC OS printer driver. If you have RISC OS 3, load a suitable printer driver according to the RISC OS 3 user guide.

   If you have RISC OS 2 double click on the `PDrivers` directory then double-click on the required printer driver. You may use other printer drivers not supplied with Hearsay, but they must be release 2 versions supporting 'fancy' text printing.

Acorn device claim protocol

Hearsay supports the Acorn device claim protocol which allows applications to share the serial port. If after loading Hearsay (or at any time while using it) the Hearsay icon on the icon bar appears shaded, it means that another application has claimed the serial port and Hearsay is inactive.

Please refer to *Appendix G* for details of how to toggle applications in and out of the active state.

Choosing a serial port

Hearsay is setup by default to use the internal serial port. If you wish to use an expansion serial port instead of the internal port, you should install a suitable device block driver.

Please refer to *Appendix H* for details of how to configure Hearsay to use the device block drivers.

Connecting a modem

Hearsay will work with most makes of modems suitable for the Acorn 32-bit RISC systems. The first thing you must do is connect your modem to the *serial port* on the computer (or expansion board). Most Acorn dealers can supply connecting leads for popular modems, and a number of typical wiring arrangements are given in the release notes. If you have any problems please contact our technical department.

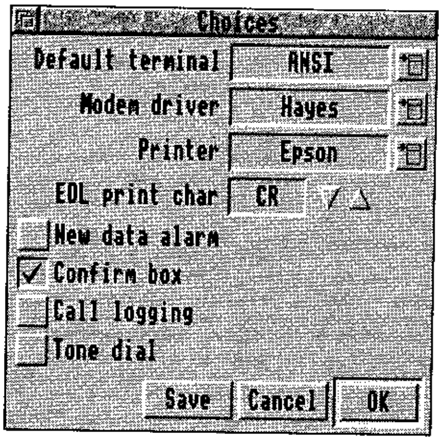
Please note that proprietary modems leads designed for PC's, will not work with some Acorn systems.

Your modem should also be connected to a standard telephone socket.

If you are connecting directly to another computer system, please refer to the release notes for a suggested wiring arrangement.

Choosing a modem driver

Hearsay is supplied with a number of modem drivers suitable for most modems. A modem driver is the interface between the computer and the modem, and controls dialling, connecting and disconnecting. To select a modem driver, choose **Choices** on the icon bar menu to open the following dialogue box.



Default terminal ANSI

Modem driver Hayes

Printer Epson

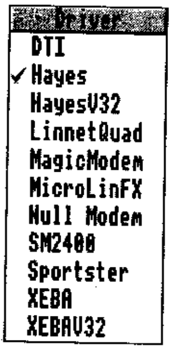
EOL print char CR

□ New data alarm

☑ Confirm box

□ Call logging

□ Tone dial

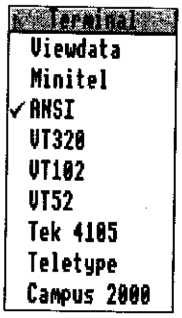


Now choose a driver from the **Modem driver** menu.

If you have a *Hayes* modem choose **Hayes**, or if you have a high-speed Hayes modem with *MNP*, choose **HayesV32**. If you are connecting directly to another computer system, choose **Null Modem**. If your modem is not Hayes compatible or not listed on the modem driver menu, please refer to the release notes for a list of modems supported and the driver suitable for each one.

To save your choice of modem driver, click on **Save**.

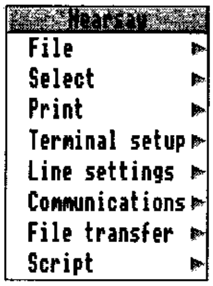
Terminal emulations



Once you have loaded a modem driver, you must choose a *terminal* emulation for the service you wish to dial. The **Terminal** menu on the icon bar menu lists the terminal emulations available in Hearsay.

The terminal you choose depends on the service you wish to use, and further guidance is given in the chapter *Icon bar menu.* At this stage however, just choose **Viewdata** for *Prestel* and other *Viewdata* services, **VT320** for *Telecom Gold,* CompuServe and other scrolling text services, or **ANSI** for most *ANSI* format *bulletin boards.* The terminal window will appear with the name of the emulation in the title bar.

Hearsay main menu

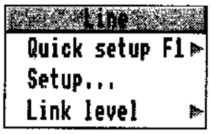


Once you have opened a terminal you can display the Hearsay main menu by clicking Menu over the terminal window.

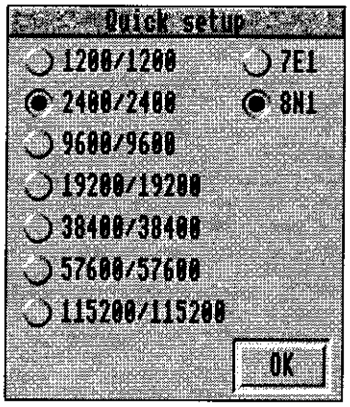
The first four options on the main menu provide facilities for controlling the operation of the terminal you have chosen. These options are introduced later in this chapter, and described in detail in the chapters *Text terminals*, *Viewdata terminal* and *Tektronix terminal.*

The last four options on the menu provide facilities concerned with the overall operation of Hearsay, and are described in chapters 4 to 7.

Baud rate and data format



Before you can call a service you must set the correct *baud rate* and *data format.* You can do this very easily using the **Quick setup** dialogue box, chosen from the **Line settings** menu.

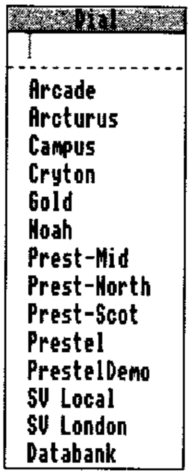


○ 1200/1200 ○ 7E1
● 2400/2400 ● 8N1
○ 9600/9600
○ 19200/19200
○ 38400/38400
○ 57600/57600
○ 115200/115200

You should choose the fastest baud rate supported by both your modem and the host system you wish to call. The two most common data formats, 7E1 and 8N1 may be chosen from this dialogue box. Click on **OK** when you have chosen the required baud rate and data format.

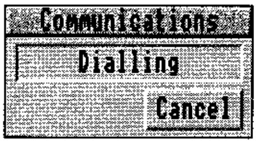
If the baud rate or data format you require is not available in the **Quick setup**, you must select them from the **Setup** dialogue box.

Dialling a number



Once you have chosen a modem driver and terminal emulator, and set the correct baud rate and data format, you may dial the number of the service you require. Enter the number you require in the **Dial** option on the **Communications** menu or icon bar menu. The number should now be dialled by the modem.

A box appears on the screen confirming that dialling is taking place, and provides a **Cancel** button on which you may click to stop dialling. Dialling is performed by a script function, so clicking on **Cancel** stops the script function and displays the message `Script stopped`.



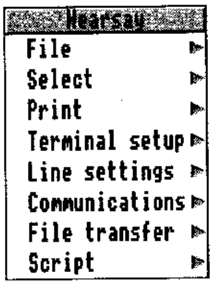
The number being dialled is normally echoed on the screen, and if your modem has a 'monitoring' feature you will be able to hear the progress of the call through the modem's loudspeaker. If the call is successful, the *host system* front page should be displayed in the terminal window, and you will be requested to *logon* and enter your password.

If you do not get connected, please check the following:

1. Is your modem connected to the computer and to a telephone socket?
2. Is the modem connecting lead wired according to the instructions given in the release notes.
3. Have you chosen a suitable modem driver?
4. Have you chosen a baud rate and data format suitable for your modem and the host system you are dialling?
5. Have you dialled the correct number and used the correct dialling code?
6. Is the number engaged? If so, click on **Dial** to try again.

Please also refer to the chapter *Communications* for further information about the **Dial** facility, and what to do if it doesn't work.

Terminal options



Once you are on line you can click Menu over the terminal window to display the main menu. A summary of the functions available from the main menu is given below, but please refer to chapters *Text terminals*, *Viewdata terminal* and *Tektronix terminal* for full descriptions.

File menu

The **File** menu allows you to save information from the terminal. Various formats are available, depending on the terminal emulation you have chosen. In the scrolling text terminals, you may save the screen, the *capture buffer* or a selected part of the buffer. In the Viewdata terminal you may save individual *frames*, or all the frames in the *tag buffer*. A *spool* option is available on this menu, which writes all incoming data to a file on disc.

Select/Action menu

In the scrolling text terminals (*VT320*, *VT102*, ANSI etc), the **Select** menu allows you to manipulate a selected block of text in the terminal. You may select text by clicking and dragging across it. Selected text may be saved, stored, printed or transmitted. In the Viewdata terminal, the **Select** menu option changes to **Action**, and allows various operations on whole frames to be performed.

Print menu

The **Print** menu allows the contents of the terminal to be printed. In the scrolling text terminals, text may be printed directly to your parallel printer, or via a standard RISC OS printer driver. In the Viewdata terminal, you can print the terminal as plain text, or via a standard RISC OS printer driver as a full graphics dump. The text dump is very quick, so it is useful if you wish to print whilst *online*.

Terminal setup menu

The **Terminal setup** menu allows all aspects of the current terminal to be configured including video and keyboard options. It provides a **Macros** facility which allows you to define macros and redefine keys on the keyboard. All settings may be saved in a script file for future re-loading. The actual options available depend on the terminal chosen, and are described in the chapter which deals with that terminal.

Line settings menu

The **Line settings** menu provides various options concerned with the communications line, including setting the baud rate and data format. The **Quick setup** option on this menu allows all the common settings to be selected from a simple dialogue box. *Link level error correction* is also controlled from this menu. Please refer to the chapter *Line settings* for full details of this menu.

Communications menu

The **Communications** menu provides a number of options for controlling the modem you have connected. Options include dialling a number, connecting and disconnecting. They are not dependant on the terminal selected. The number directory, described later in this chapter, is also selected from this menu. This allows you to store all the numbers you wish to dial in a directory for easy recall. Each entry is given a name, together with other information allowing Hearsay to automatically dial the number, go online and enter logon and password strings for you. Please refer to the chapter *Communications* for full details.

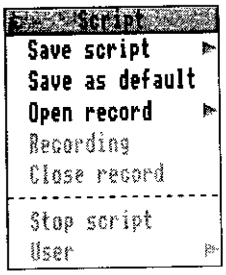
File transfer menu

The **File transfer** menu provides several *file transfer protocols* allowing you to transfer files over the communications link. You may receive files from, or send files to, host systems, bulletin boards or other users. Hearsay provides a range of file transfer protocols which should allow you to communicate with almost any system, they include: *Xmodem*, *Ymodem*, *Zmodem* and *Kermit*. Please refer to the chapter *File transfer* for full details.

Script menu

An important feature of Hearsay is its integrated script language called CScript, which is based on a subset of the language C. CScript programs may be written using a text editor such as Edit, and executed by double-clicking on them. The **Script** menu provides a number of options associated with script files. These include saving the current configuration as a script file, stopping a script file which is running, and recording a logon procedure in script file format. Further details of these options are given in the chapter *Script menu*, and a full description of the CScript language is given in the chapter *CScript*.

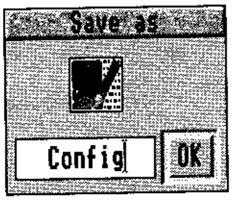
Saving the configuration



Whilst this chapter is only intended as an introduction to Hearsay, it should have provided enough information for you to configure Hearsay and get online. Therefore at this stage, you may wish to save the current configuration so that you can easily return to it in the future.

If you choose **Save as default** on the **Script** menu, your current configuration will be saved in a script file which is automatically executed when Hearsay is run. This allows you to customise the Hearsay start-up configuration.

More usefully, **Save script** on the **Script** menu allows you to save the current configuration in a named file.

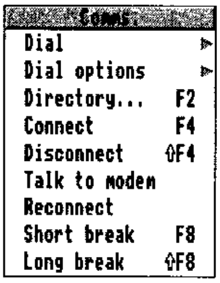


You can re-load this configuration at any time by double-clicking on the script file.

The settings saved using these options include:

- baud rate, data format etc.
- all terminal configuration
- default file transfer protocol
- file transfer protocol configuration
- link level error correction configuration
- auto-redial configuration
- dial prefix status

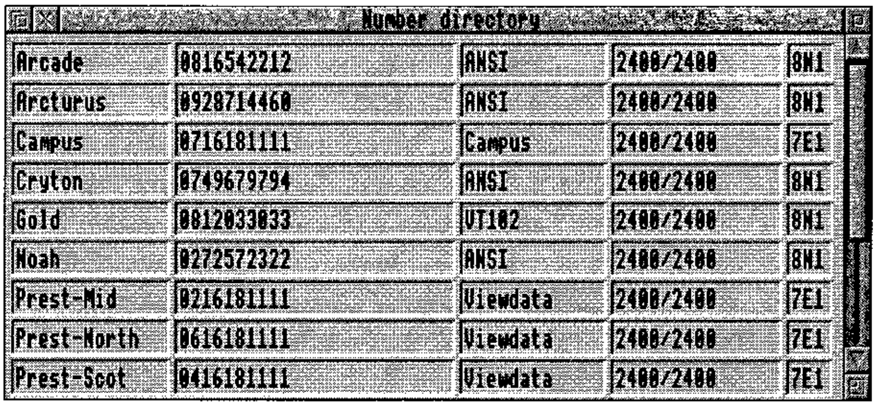
Number directory



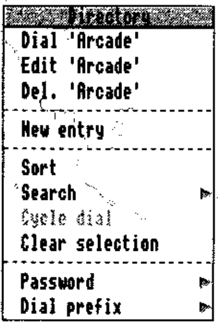
Earlier sections of this chapter have described how to set the baud rate, parity etc and dial a number to get online. An easier method of doing this is to use the number directory to store all the numbers you need to dial. Each entry in the directory is given a name, and contains all the information needed to *auto dial* the number, select the correct terminal and logon to the service.

This section describes just the basic features of the number directory; a full description is given in the chapter *Communications*.

Choose **Directory** on the **Communications** menu to display the number directory window.

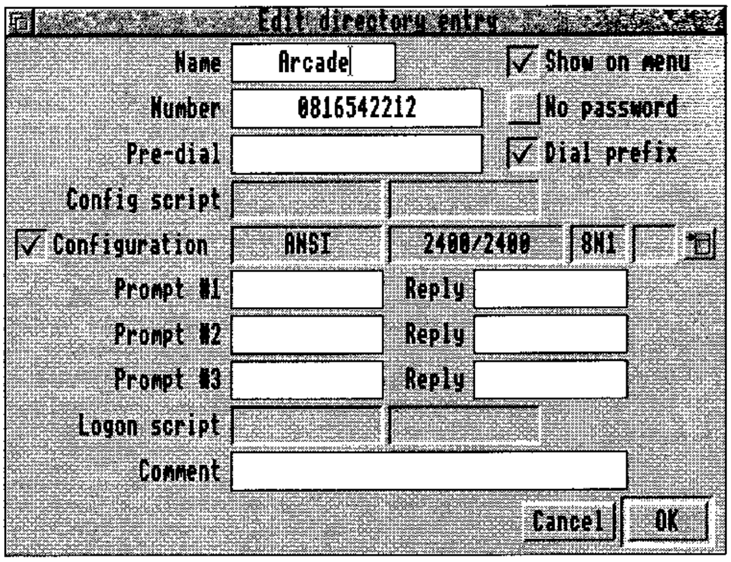

Number directory

Arcade	0816542212	ANSI	2400/2400	8N1
Arcturus	0928714460	ANSI	2400/2400	8N1
Campus	0716181111	Campus	2400/2400	7E1
Cryton	0749679794	ANSI	2400/2400	8N1
Gold	0812033033	VT102	2400/2400	8N1
Noah	0272572322	ANSI	2400/2400	8N1
Prest-Mid	0216181111	Viewdata	2400/2400	7E1
Prest-North	0616181111	Viewdata	2400/2400	7E1
Prest-Scot	0416181111	Viewdata	2400/2400	7E1



This window lists all the entries in the directory. Use the vertical scroll bar to display any entries not in view. The name, number, terminal, baud rate and data format is given. Click Menu over an entry to highlight that entry and display the menu shown.

From this menu you may dial, edit or delete the highlighted entry or add a new entry.



Edit directory entry

Name: Arcade ☑ Show on menu

Number: 0816542212 □ No password

Pre-dial: ☑ Dial prefix

Config script:

☑ Configuration: ANSI 2400/2400 8N1

Prompt #1: Reply:

Prompt #2: Reply:

Prompt #3: Reply:

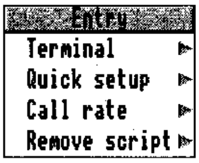
Logon script:

Comment:

To add a new entry choose **New entry** to display the edit entry dialogue box. Please note that the new entry is based on the currently highlighted entry.

To add a new entry proceed as follows:

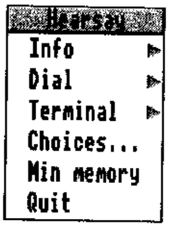
1. Enter the name in the **Name** icon.
2. Enter the telephone number in the **Number** icon.
3. Make sure that the **Configuration** icon is selected.
4. Click Menu over the arrow icon on the **Configuration** line to display the **Entry** menu shown.



5. Use the **Terminal** and **Quick setup** options on this menu to set the required terminal emulation, baud rate and data format. Your settings will be displayed in the main dialogue box.
6. You do not need to set anything else at this stage, but make sure that all the other writable icons in this box are blank.
7. Click **OK** to close the box and update the directory with the new entry. Click **Cancel** if you do not want to add the entry to the directory.

The new entry will be added to the bottom of the list of numbers in the directory window, and may be dialled by clicking Menu over it and choosing the **Dial** option. When you close the number directory window the directory will be updated on disc for future use. Your entry name will also be available from the **Dial** option on the **Communications** menu and the icon bar menu.

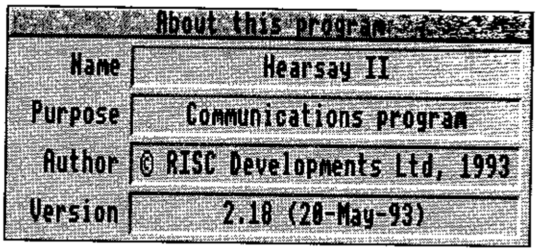
3 Icon bar menu



To display the Hearsay icon bar menu click **Menu** on the application icon on the icon bar at the bottom of the screen.

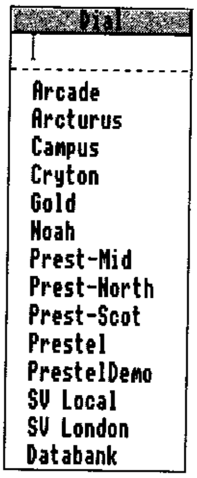
Info

Info displays a dialogue box giving information about the version of Hearsay you are using. Please quote the version number given in any correspondence with RISC Developments about this product.


About this program

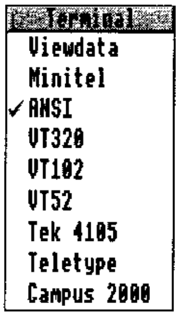
Name	Hearsay II
Purpose	Communications program
Author	© RISC Developments Ltd, 1993
Version	2.18 (28-May-93)

Dial



This option allows you to dial a service. You can either enter the telephone number at the caret provided, or choose the name of the service you require from the menu. The names in the menu relate to the names of the entries in the telephone directory.

This menu is identical to the **Dial** sub-menu on the **Communications** menu, so please refer to the chapter *Communications* for further details.



Terminal

The **Terminal** menu allows you to choose the terminal type you wish to open.

Viewdata

The Viewdata terminal is used to communicate with Prestel and other similar page-orientated systems which use the Viewdata standard. It is a 7-bit terminal utilising teletext text and graphics in 7 colours. Full details of this terminal are given in the chapter *Viewdata terminal.* This terminal is also referred to as CEPT profile 3.

Minitel

Minitel is the standard French Viewdata system often referred to as CEPT profile 2. It's operation is similar to Viewdata above, but it offers higher resolution graphics, more colours and extended character sets. The differences in operation between the Viewdata and Minitel terminals are given in the chapter *Viewdata terminal.*

ANSI

The ANSI terminal is very similar to the VT102 terminal, except that it supports the ANSI colour *escape sequences* as used by the ANSI.SYS screen driver on an IBM PC. In addition to colour, this terminal features the complete IBM PC character set. This terminal should be used when connecting to systems which offer 'ANSI colour'. Operation is identical to the VT320 terminal, and is described in the chapter *Text terminals.*

VT320

The VT320 terminal is the most up-to-date terminal provided by Hearsay and should be used if your host system supports VT320 or VT220 terminals. It is an 8-bit version of the very popular VT102 terminal, but offers an international character set, downloadable function keys, and additional editing commands. The VT320 terminal makes use of the 25th line at the bottom the the terminal window used as a status line during VT220 and VT102 operation. Operation of this terminal is described in detail in the chapter *Text terminals.*

In order to make full use of the VT320 terminal, the **8-bits** option on the **Line mode** dialogue box must be set.

VT102

This terminal is a complete emulation of a DEC VT100/VT102 terminal with advanced video option. If your host computer offers you a choice of VT100 or VT102, you should choose the latter since the VT102 has additional commands that speed up text editing. Operation of this terminal is described in detail in chapter *Text terminals.*

VT52

This is the most primitive of the VT terminals in common use. If at all possible you should choose VT102 instead of this, but if the host computer does not support VT102, the *VT52* terminal may prove a useful lowest common denominator. Whilst the VT52 terminal is not as sophisticated as the other text terminals, it's operation is almost identical and is described in the chapter *Text terminals.*

Tek 4105

This terminal emulates the Tektronix 4010, 4014 and 4105 graphics terminals. Full details of the operation of this terminal are given in the chapter *Tektronix terminal.*

Teletype

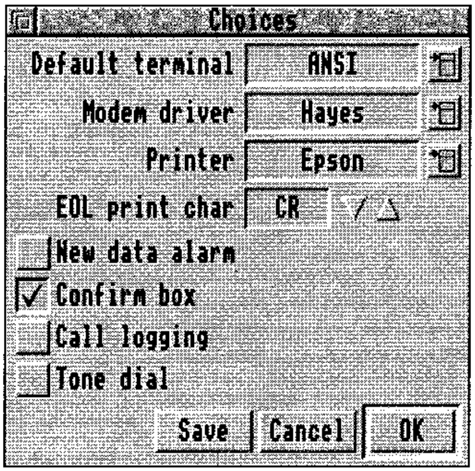
This is a very simple 80 column scrolling terminal supporting a very limited range of control characters, and no escape sequences. It may prove useful for debugging or connection to systems which send *control codes* that upset the more sophisticated terminals. Operation is similar to the VT320 terminal, and is described in the chapter *Text terminals.*

Campus 2000

The Campus 200 terminal is suitable for connecting to Campus 2000 or TTNS. It provides a Viewdata terminal which may be switched under control of the host system to a VT102 terminal, and vice versa. Operation of this terminal is identical to the text and Viewdata terminals described in the chapters *Text terminals* and *Viewdata terminal.*

Choices

The **Choices** dialogue box contains a range of user-selectable preferences which control the operation of Hearsay.



Default terminal ANSI

Modem driver Hayes

Printer Epson

EOL print char CR

□ New data alarm

☑ Confirm box

□ Call logging

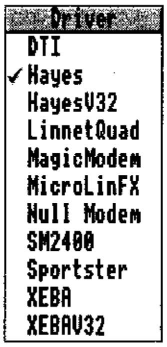
□ Tone dial

Default terminal

Clicking Select on the Hearsay icon on the icon bar always re-opens the previously opened terminal window. If there is no previous terminal e.g. immediately after running Hearsay or after using **Min memory**, the default terminal is opened.

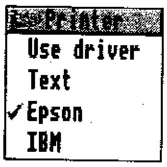
Default terminal allows you to choose the default terminal.

Modem driver



This menu lists the modem drivers available on the system and allows you to choose one suitable for your modem. For more details about choosing modem drivers please refer to the chapter *Getting started.*

Technical information, for those users who wish to write their own modem drivers is given in *Appendix F*.



Printer

If **Use driver** is selected, text printing is carried out via a standard RISC OS printer driver which must be installed on the icon bar. The 'fancy' text printing facilities of release 2 and 3 printer drivers are utilised to print bold, italics etc.

If **Epson** is selected, text printing is carried out via the normal printer port. Standard Epson escape sequences are used to print bold, italics etc.

IBM is similar to **Epson** above, but prints the full ANSI character set in the ANSI terminal. You will have to consult your printer user guide to find out if your printer is compatible with the full IBM character set.

If **Text** is selected, only plain ASCII text is printed and styles such as bold and italics are ignored.

EOL print char

This option defines the character which is sent to the printer at the end of each line when printing. It may be set to **CR**, **LF**, **CRLF**, **LFCR** or **Raw**. If **Raw** is selected, the original end-of-line character is printed, unaltered.

New data alarm

If this option is selected and there is no terminal window open, the Hearsay icon on the icon bar will flash if data is received. It allows you to close a terminal window and use other applications whilst you are waiting for the host system to respond. Typically it is used when connected directly to another computer carrying out time consuming work.

Confirm box

This option allows you to enable or disable the confirmation prompts that are used throughout the package. If it is selected, you will be prompted for confirmation whenever an operation is carried out that cannot be undone and would cause the loss of data e.g. deleting a buffer.

Call log

This option allows you to enable and disable call logging. If call logging is enabled, each call that you make is logged in an *ASCII* file called `LogFile` located in the `!Hearsay` application directory. Each log entry gives the time the call was made, the telephone number or directory entry called, the time the call was terminated and the number of seconds online. If the call was made using the number directory and a call rate was specified, the number of units used is also displayed.

Call logging allows you to find out what calls have been made and to whom, and from this information the cost of the calls can be calculated. The log file can be examined by loading it into Edit or a word processor.

A typical log file might look like this:

```
10:02am 24 Jun 91 : Online to Prestel
10:03am 24 Jun 91 : Offline to Prestel after 132
12:15pm 24 Jun 91 : Online to 016181111
12:34pm 24 Jun 91 : Offline to 016181111 after 465
18:01pm 25 Jun 91 : Online to Gold
19:11pm 25 Jun 91 : Offline to Gold after 795 used 6 units
```

The default name `LogFile` can be changed by editing the system variable `Hearsay$LogFile` which is defined in the Hearsay `!Run` file.

Tone dial

If **Tone dial** is selected, numbers will be tone dialled instead of pulse dialled. Please note that tone dialling is only possible if your modem and local telephone exchange support it.

Save

This option saves the settings you have specified for the above options. If you do not save your choices, they will be valid only for the current session. If you save them, they will be loaded and set automatically next time Hearsay is run.

Min memory

Normally if you close a terminal in Hearsay, the contents of the window are retained in memory. If the terminal is re-opened by clicking on the Hearsay icon on the icon bar, the contents remain intact. This also applies to the contents of any buffers, such as the scrolling text capture buffer and the Viewdata tag buffers. The memory used to store this information is reclaimed when you quit Hearsay, or when the **Min memory** option is chosen. **Min memory** is useful if you wish to keep Hearsay installed on the icon bar, but reclaim as much memory as possible for another task.

Min memory is also used to flush the print buffer when printing text using the RISC OS printer drivers. Further details of this feature are given in the description of the print options later in this user guide.

Quit

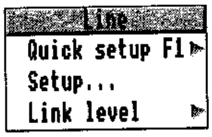
Quit exits Hearsay. You will be warned if any Viewdata buffers are modified and unsaved, and given the opportunity the save them, continue using Hearsay, or to quit. This option also closes any open spool files.

If you quit Hearsay when online or when a file transfer is taking place, a dialogue box will be displayed asking for confirmation (see *Appendix G* for further details).

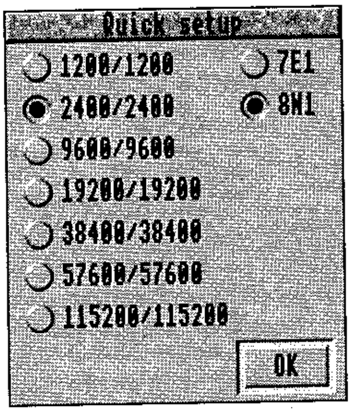
4 Line settings

The **Line Settings** menu on the main menu allows you to set the baud rate and data format, as well as various other line settings.

Quick setup



This option opens a dialogue box listing the most commonly used baud rates and word formats. If the baud rate or data format is not shown, you must choose them from the **Setup** dialogue box (described below).



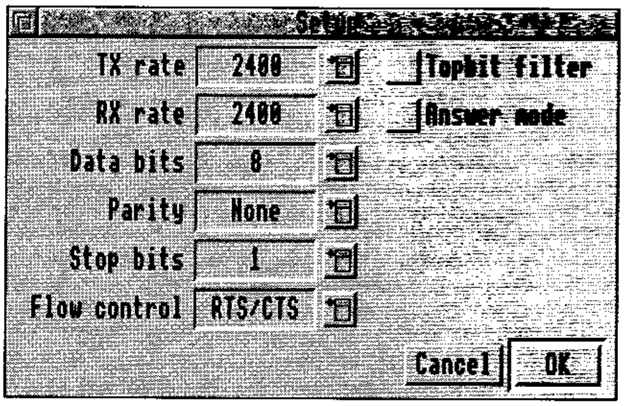
○ 1200/1200 ○ 7E1
● 2400/2400 ● 8N1
○ 9600/9600
○ 19200/19200
○ 38400/38400
○ 57600/57600
○ 115200/115200

The baud rate standards are listed using the CCITT convention e.g. V22, followed by the actual communications speed. You should choose the fastest speed supported by both your modem and the host system to which you are connecting.

The two most common data formats, 7E1 and 8N1 may also be chosen from this dialogue box; again, choose the one that is suitable for the host system to which you are connecting.

Setup

The **Setup** dialogue box allows you to setup all baud rates, word formats and other line settings. It must be used if the settings you require are not available on the **Quick setup dialogue box.**



Setup

TX rate 2400

RX rate 2400

Data bits 8

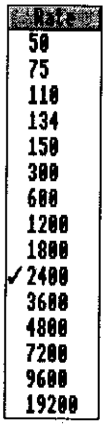
Parity None

Stop bits 1

Flow control RTS/CTS

□ Topbit filter

□ Answer mode

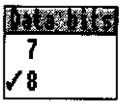


TX rate

The **TX rate** menu allows you to choose the transmit baud rate. The menu lists all the baud rates possible on Acorn RISC systems. You need only use this menu on the rare occasion when the the baud rate you require is not available on the **Quick setup** dialogue box.

RX rate

The **RX rate** menu allows you to choose the receive baud rate. The menu lists all the baud rates possible on Acorn RISC systems. You need only use this menu on the rare occasion when the the baud rate you require is not available on the **Quick setup** dialogue box.

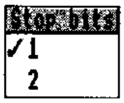


Data bits

Data bits may be set to **7** or **8.**

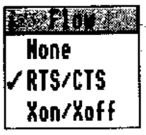
Parity

Parity may be set to **None, Even, Odd, Mark** or **Space.**



Stop bits

Stop bits may be set to **1** or **2.**



Flow control

This sub-menu allows you to choose the type of *flow control* you wish to use. Flow control is the mechanism which prevents one end of a link transmitting data when the other end is not ready to receive it.

If **None** is selected, flow control is disabled.

RTS/CTS selects hardware flow control, and should be chosen if you are communicating via a modem. **RTS/CTS** flow control will only work if the *RTS/CTS* serial connections are made according to the wiring instructions given in the release notes

Xon/Xoff switches on the software *Xon/Xoff* flow control. The Xon/Xoff protocol controls the flow of data by sending the Xoff character (ASCII 19, Ctrl S) to stop transmission of data, and the Xon character (ASCII 17, Ctrl Q) to resume.

Topbit filter

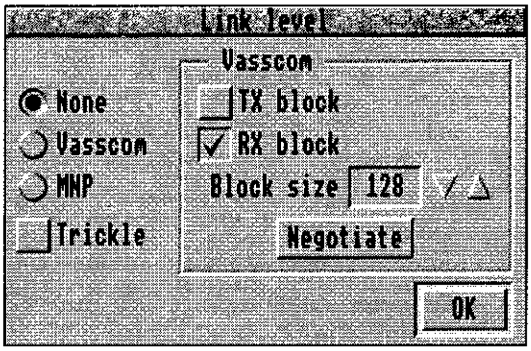
If **Topbit filter** is selected, the 8th-bit of characters greater than ASCII 127 is masked out. So 128 becomes 0, 129 becomes 1 etc. Try using this option if the text you receive is corrupted with top-bit accented characters.

Answer mode

In any communications link via a modem, one end of the link must be in *originate mode*, and the other end in *answer mode*. These modes set the frequency for the data communication. Normally when connecting to most systems you should be in originate mode (since you are the originator of the call) i.e. answer mode should not be selected. If you are running a host system or communicating with another modem which is in originate mode, you must select **Answer mode**.

Link level

The **Link level** menu controls link level error correction. Link level error correction ensures all data sent and received is error free. It does this by transferring data in blocks with a checksum value at the end of the block. If the checksum shows that the data is corrupted, then the block is automatically re-transmitted. Since link level error correction slows down the transfer of data, it is really only useful when communicating at high baud rates, where errors are most likely to occur. At slow speeds, it is usually unnecessary since errors are infrequent, and the time taken to re-transmit a block can be more annoying than the odd corrupted character. Hearsay provides two error correction protocols: Vasscom and MNP.



● None
○ Vasscom
○ MNP
□ Trickle

Vasscom
□ TX block
☑ RX block
Block size 128

None

This options switches off all link level error correction.

Vasscom

Vasscom is the error correction protocol used by Prestel. It is intended for use at higher baud rates, but will work at all speeds. To use Vasscom you must set the data format to 8N1. If **Vasscom** is selected before you dial a number, then it will be automatically negotiated when the modem connects. However, you can start Vasscom error correction at any time, by choosing **Vasscom** then clicking on **Negotiate**.

Vasscom can be configured with the following options.

If **TX block** is selected, transmitted data is error corrected. By default this option is not selected.

If **RX block** is selected, received data is error corrected. By default this option is selected.

Block size allows you to change the size of the blocks in which the data is transferred.

Smaller blocks cause data to be transferred more slowly overall, but are more efficient on a noisy line. Large blocks are the most efficient on a good line, but cause a long delay if a block has to be re-transmitted. The default block size is 128.

Negotiate establishes a Vasscom link with the host system, and should be used after selecting **Vasscom**. In this case Hearsay automatically negotiates with the host. **Negotiate** should also be used if you change the **RX block**, **TX block** or **Block size** settings whilst Vasscom is switched on.

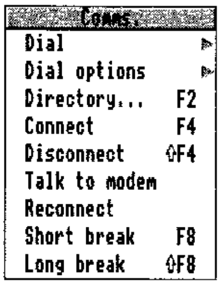
MNP

This option selects MNP link level error correction. If your modem supports MNP, and if an appropriate modem driver is selected, your modem's internal error correction is chosen, otherwise software MNP level 2 is chosen. Normally it is better to use the modem's error correction, since it will probably support a more efficient level, such as 3, 4 or 5. Hearsay will try and establish an MNP connection for a couple of seconds directly after your modem has connected, so you must choose MNP before dialling a number.

Trickle

Trickle mode may be used with Vasscom or MNP, and smooths out the appearance of data being received in the terminal. It works by displaying received data at the current baud rate instead of in bursts when each block arrives.

5 Communications

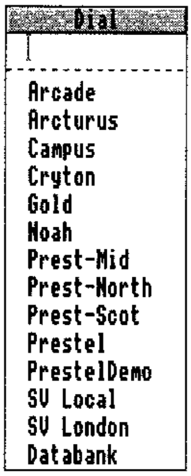


The **Communications** menu provides a number of options concerned with controlling your modem. These options include dialling a number, connecting and disconnecting the modem.

Dial

This option allows you to to call a telephone number. You can either enter the telephone number at the caret provided, or choose the name of the service you require from the menu.

The names in the menu relate to the names of the entries in the number directory (see later in this chapter for further details). If the directory is password protected, the first time in a session that you dial a number from the directory you will be asked to enter the password. Failure to enter the correct password will prevent the number being dialled.



If you enter a number, remember that some *PABX* systems require a prefix (e.g. 9) to be added to the number to dial an 'outside' line. You may add the prefix to each number you dial individually, or use the **Dial prefix** option described later, which prefixes each number dialled with a specified code. Often a pause is needed between dialling the prefix and the main number. The pause character is usually a , (comma), but some intelligent modems may use a different character, so consult your modem handbook for further details. Again, the pause character may be added directly to the number, or specified globally using the **Dial prefix** option.

To re-dial the last number called, simply click on **Dial**, there is no need to re-enter the number or re-select the name.

Please note that the **Dial** option will only work if your modem is capable of auto-dialling, and that a suitable modem driver has been selected. If it does not feature auto-dialling, you will have to dial manually and **Connect** when you hear the high pitched *carrier* tone.

Call problems

As you probably realise from your own experience with the telephone network, making a call can be subject to a number of difficulties. The most common problem is that the number is engaged. If your modem has line monitoring, you will be able to hear exactly what is happenıng, and some intelligent modems will display messages on the screen if problems occur. If your modem has neither of these features, it may be useful to have a telephone plugged into the same line, so that you can monitor the progress of the call. Always wait until the number has been dialled before lifting the handset.

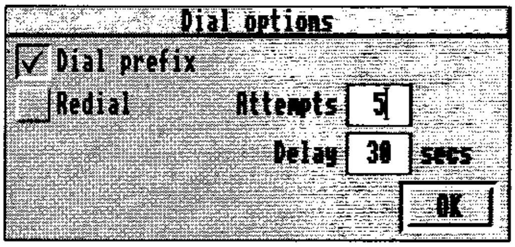
If you have problems logging on to larger databases which have multiple telephone lines, first check that you have selected the correct modem driver and the number that you are dialling is correct. Finally, check the connections between the computer and modem, and modem and telephone line.

If you go online, but the data received is corrupted, check that the baud rate and word format are set correctly for the service you have called.

As a last resort you should connect a telephone to the same line (special adaptors are available to do this), and dial the number manually. When you hear the high-pitched carrier tone from the host computer, select **Connect** and then replace the receiver.

Dial options

This option opens the following dialogue box.



☑ Dial prefix

□ Redial Attempts 5

Delay 30 secs

Dial prefix

This option determines whether the dial prefix specified in the number directory is appended to the start of any number dialled. A dial prefix is useful if you are dialling from a private exchange, and need to prefix the normal telephone number with a code, usually 9. You can put the normal telephone number in the directory entry, and set the dial prefix to 9.

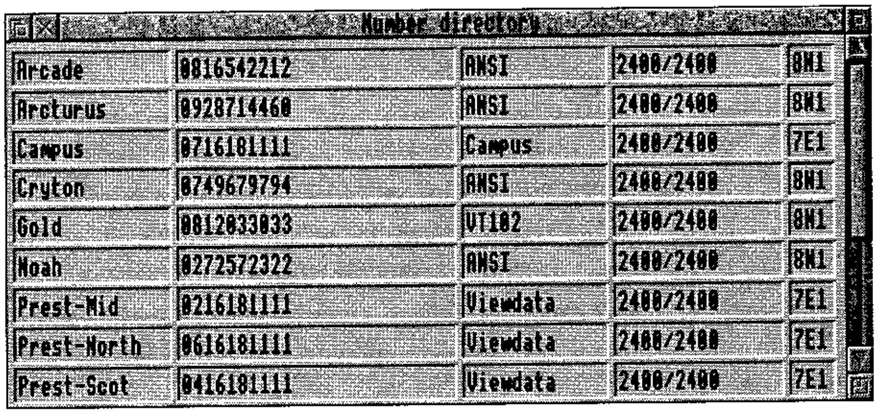
It is also useful if you are using a telephone system such as Mercury which requires a string of digits to be sent before the number. You can also enable or disable dial prefixes for individual numbers in the directory using the **Dial prefix** option in the **Edit** dialogue box. This allows you to dial certain numbers with prefixes, and others without.

Redial

If **Redial** is selected, the **Dial** option described above will automatically dial the chosen service a specified number of times until the call is successful. The **Attempts** and **Delay** options allow you to specify the number of attempts that should be made to dial the number, and the delay in seconds between each attempt.

Directory

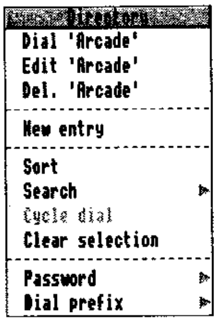
The **Directory** option opens the Hearsay number directory. This directory allows you to store all the numbers you wish to dial, in a file for easy recall at a later date. Each entry is given a name, together with other information which allows Hearsay to dial the number, enter the correct terminal, set the baud rate, respond to prompts on the host system, and send logon sequences and passwords. Additionally, it can automatically execute script files both before and after dialling the numbers. The directory may be password protected, preventing unauthorised users dialling or examining directory entries.



Number directory

Arcade	0816542212	ANSI	2400/2400	8N1
Arcturus	0928714460	ANSI	2400/2400	8N1
Campus	0716181111	Campus	2400/2400	7E1
Cryton	0749679794	ANSI	2400/2400	8N1
Gold	0812033033	VT102	2400/2400	8N1
Noah	0272572322	ANSI	2400/2400	8N1
Prest-Mid	0216181111	Viewdata	2400/2400	7E1
Prest-North	0616181111	Viewdata	2400/2400	7E1
Prest-Scot	0416181111	Viewdata	2400/2400	7E1

Choose **Directory** to open the number directory window, listing each entry in the directory. If the directory is password protected, you will be prompted for the password before the directory is opened. Each entry in the window is split into 5 fields; entry name, telephone number, terminal emulation, baud rate and data format. The last three fields will be blank if they have not been set explicitly for that entry (in which case they may be set by any pre-dial script files or to the start-up defaults). Double-clicking Select on any number causes it to be dialled. Double-clicking Adjust on any number opens the **Edit** dialogue box for that number.



Click Menu over the directory to display the menu shown.

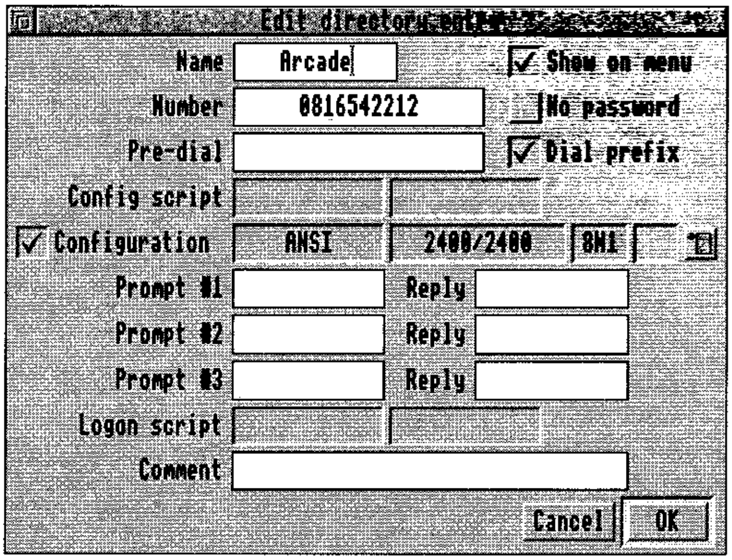
Many of these menu options apply to the selected entry or entries. To select a single entry click Select over it. To select a number of entries, click Adjust over each one. If there is no selected entry when you click Menu, the entry under the mouse pointer is selected.

Dial

This option dials the selected entry. It is greyed out if there is more than one entry selected.

Edit

Edit opens a dialogue box for editing the selected entry. This option is greyed out if there is more than one entry selected.



Edit directory entry

Name: Arcade

Number: 0816542212

Pre-dial:

Config script:

☑ Configuration: ANSI 2400/2400 8N1

Prompt #1: Reply:

Prompt #2: Reply:

Prompt #3: Reply:

Logon script:

Comment:

☑ Show on menu

□ No password

☑ Dial prefix

Name is the name of the directory entry.

Number is the telephone number to be dialled. This may be left blank if you are connecting directly to the host system and not using a modem.

Pre-dial is an optional string which is sent to the modem before dialling. Typically this is used to configure your modem for the number you are dialling. This string may contain the standard escape sequences |A, |B etc, listed in the chapter *Macros*.

If **Show on menu** option is selected, the name of this entry will appear on the **Dial** menu, otherwise it will be omitted. It is useful if you create a very large directory of numbers, but only want a selection to appear on the **Dial** menu.

If **No password** is selected, the entry will not be password protected. This option is useful if your directory is password protected, but you wish to allow this number to be dialled by users who do not know the password.

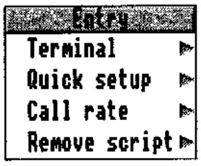
If **Dial prefix** is selected, it enables a dial prefix to be appended to the start of the telephone number when it is dialled. The dial prefix will only be used if the **Dial prefix** option on the **Dial options** dialogue box.

The two boxes labelled **Config script** are used to specify script files that are to be executed immediately before the number is dialled. To specify a script file, drag its icon into one of the boxes. Typically one script file would normally be a configuration file and the other a key macro file. This feature allows the system to be configured precisely for each individual entry in the telephone directory.

If **Configuration** is selected, the directory entry is configured according to the settings shown. These settings are the minimum configuration needed to get online, and are used to set up directory entries quickly, without getting involved with configuration files. They override settings specified in any pre-dial script files.

If **Configuration** is not selected, the directory entry will use the settings specified in any pre-dial script files, or the default settings if no pre-dial script files are specified.

Click Menu to display the menu shown, which allows the following configuration settings to be specified:



Terminal allows you to choose which terminal emulator will be entered if the call is successful.

Quick setup opens the standard dialogue box for specifying the baud rate and word format.

Call rate allows you to specify one of 5 call bands which relate to the distance of the call being made. These bands are used to calculate the number of units used since going online. The units used are displayed on the optional status line of the scrolling text terminals, or the keypad of the Viewdata terminal. The allowable call bands are:

`None`	no units are displayed
`L`	local calls
`a`	calls up to 35 miles (56.4km)
`b`	calls over 35 miles
`bl`	calls at the reduced rate over 35 miles
`m`	calls to mobile phones

Remove script allows you to remove any of the pre-dial or post-dial script files attached to the entry. A menu is displayed listing the names of any script files specified. Simply click on any you wish to remove.

The three sets of **Prompt/Reply** boxes are used to provide a simple means of automatic logon. After the call has been successfully connected, Hearsay waits for the first prompt, and then transmits the first reply. It then proceeds to the next prompt. If no prompt strings are specified, the reply strings are transmitted immediately after the call has connected. The example shown in the dialogue box on page 33 is a typical logon for the Prestel demonstration database. Both **Prompt** and **Reply** strings may contain the standard escape sequences `|A`, `|B` etc. A full list of these is given in the chapter *Macros*.

The two boxes labelled **Logon script** are used to specify script files that are to be executed after dialling. To specify a script file, drag its icon into one of the boxes. Typically, post-dial script files are used for complicated logon procedures, or automatic downloading of mail etc.

The **Comment** field is for users comments about the directory entry.

Click on **OK** to save the edited entry into the directory, or **Cancel** to cancel all the changes made.

Del.

Del. deletes the selected entry. You will be prompted for confirmation before deletion takes place (only if **Confirm box** is selected on the **Choices** dialogue box). If more than one entry is selected, this option changes to **Delete selection** on the menu, and all selected entries are deleted.

New entry

New entry creates a new entry in the directory based on the selected entry (except for the **Name** and **Number**). If there is no selected entry, a completely blank entry is created. A new entry is edited in exactly the same way as described for the **Edit** option above.

Sort

This option sorts the directory entries into alphabetical order according to their names. There may be a short delay during sorting, especially for directories with a large number of entries.

Search

This option searches the name field of the for the specified string. If it is found, the entry is displayed and highlighted. The search commences from the start of the directory, or from a selected entry. After an entry has been found, the search may be continued with a further click on **Search**. You can use * in the search string to specify a wildcard character which will match any string of characters. The search is not case specific i.e. upper and lower case characters are treated the same.

Cycle dial

This option is only available if more than one directory entry is selected. It dials each number in the selection in turn until a connection is made.

Clear selection

This option de-selects any selected entries.

Password

This option is used to specify a password to prevent unauthorised dialling or examination of directory entries. If a password is specified, the first time in a session that you dial a number or open the directory, you will be asked to enter the password. Password protection may be disabled for individual entries by selecting the **No Password** option in the **Edit** dialogue box.

Note that both the contents of the directory and the password are encoded to prevent unauthorised reading directly from disc. This means that in the event of the password being forgotten it is impossible to restore it. In this situation you must delete the directory called `TeleDir` stored in the `!Hearsay` application directory, so that next time Hearsay is run, a new blank directory is created.

Dial prefix

This option allows you to specify a dial prefix which may be appended to the start of any numbers that are dialled. The option **Dial prefix** on the **Dial options** dialogue box determines whether the prefix is used or not. Typically the dial prefix will be blank for most users, 9 if you are dialling from a private exchange, or a user ID if you are using the Mercury telephone system. Please note that since the dial prefix is stored in the telephone directory, it too may be password protected using the Hearsay password system described above.

Connect

This is used to put the modem online. Typically it is used to put the modem online after establishing voice communication with someone over the telephone.

Exactly what happens when **Connect** is selected, depends on the current modem driver. The null modem driver is always online, so clicking on connect will just re-initialise the serial system and go online. Most modem drivers however, will go into their connect sequence. If a carrier from another modem is detected, then the modem will go online, otherwise the connect sequence will fail and the modem will remain offline.

When connecting to another modem in this way, it is important that one of the modems is in answer mode and the other in originate mode. If you have an intelligent modem, the mode in which your modem will attempt to connect is set on the **Line settings** menu. For less intelligent modems a knob on the front of the modem must be used to select answer or originate before connection is attempted. In general, it doesn't matter whether you choose to be in answer or originate mode, as long as the remote modem is in the opposite mode.

If you are using a 1200/75 modem, use the **Quick setup** dialogue box to set the correct mode; V23O is originate mode and V23A is answer mode. The modem sending the majority of the data should be in the mode with the fastest baud rate. Normally this is the host system, so you should be in originate mode. But, if you are sending a file to someone, you should be in V23A answer mode; transmit rate 1200, receive rate 75.

Disconnect

This is the opposite of **Connect**, it will attempt to put the modem *offline*. For simple modems, it will do just this dropping the phone line immediately. For intelligent modems, the hang up sequence (defined in the modem driver), will be sent to the modem and this may take a few seconds to take effect.

Talk to modem

If you have an intelligent modem then after successfully dialling or connecting to a remote computer, everything you type to the modem, will go to the remote computer. However, occasionally, you may wish whilst online, to send commands to the modem. Normally an escape sequence is provided to allow you to do this for example on Hayes modems it is a pause followed by +++ followed by another pause. After sending this escape sequence, you can 'talk' to the modem e.g. send it AT commands. The **Talk to modem** option simply sends this escape sequence.

Reconnect

After selecting **Talk to modem**, you may wish to continue with your session on the remote computer. The **Reconnect** option sends the necessary command to the modem.

Short break

Short break transmits a break level on the serial port data output for a few tenths of a second. The usual use of **Short break**, is to 'wake up' the remote computer i.e. prompt them into talking to you. You may need to find out if this applies to the systems you usually use, typically mainframe, minis or local area networks.

Long break

Long break transmits a break level on the serial port data output for a few seconds.

6 File transfer

Hearsay provides several file transfer protocols allowing you to transfer files between computers over a communications link.

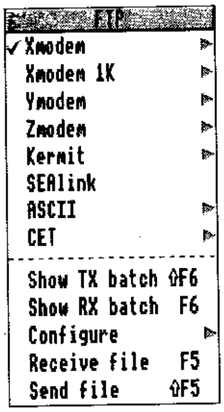
First though, why are file transfer protocols needed, and why is there more than one? When you exchange a file with another computer, you want as much of the original file as possible to be preserved. The actual contents of the file should not be altered; despite the fact that it may be transmitted over noisy telephone lines. So the first reason for having a File Transfer Protocol (FTP) is to ensure there are no errors in the transferred file. Secondly, transferred files, should not change in length and filenames should be preserved. Other file attributes like date stamps might also be transmitted with the file. The final reason for FTP's, is to provide a structured framework in which the file can be transferred, so that each computer knows exactly how the transfer will take place.

Originally, files were transferred using the ASCII protocol, in which one computer typed out the file to the other which saved all received data to file. The disadvantages with this are that there is no error correction, and no file information is transmitted; making it necessary for the receiver to spot the end of the file and stop spooling. Another disadvantage, is that files can only contain ASCII codes, and many files contain 8 bit data e.g. Basic programs. Many communications channels are only 7 bits wide, and if only 7 bits of data can be transmitted or received, then special measures must be taken if 8 bit files are to be transferred successfully. Similarly, many computers will be upset if control codes in the range 0-31 are sent to them.

Hearsay provides a wide range of file transfer protocols that should allow you to transfer files between your computer and almost any host system.

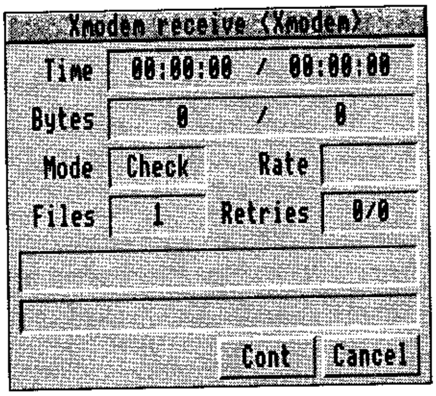
Xmodem	The most widespread FTP, but slow.
Xmodem 1k	Improved Xmodem for good communications lines.
Ymodem	Widely available *batch* transfer.
Zmodem	Fast and reliable batch transfer.
Kermit	Universal batch transfer. Reliable but slow.
SEAlink	Uncommon batch transfer - faster then Xmodem.
ASCII	Plain ASCII text transfer. No error correction.
CET	Frame based FTP used in Viewdata systems

Receiving files



To receive (or *download*) a file you must first choose a file transfer protocol from the **File transfer** menu. You should choose one that is available on your host system. If your host system offers a choice of protocols, then consult the descriptions of the protocols later in this chapter, to choose the most suitable.

Next, follow the instructions on the host system to initiate the download, and click on **Receive file**. During downloading the file transfer status box is displayed, giving information about the progress of the transfer.


Xmodem receive (Xmodem)

Time 00:00:00 / 00:00:00

Bytes 0 / 0

Mode Check Rate

Files 1 Retries 0/0

Time gives the time that has elapsed so far for the file being transferred, and the estimated total for the entire file. The time is not displayed for Xmodem, Xmodem 1K and ASCII transfers, because the file length is not transmitted along with the file, and a total time cannot be estimated.

Bytes gives the number of bytes received, and the total length of the file (not for Xmodem, Xmodem 1K and ASCII transfers, since this information is not transmitted with the file). For CET transfer this option changes to **Blocks**, and shows the number of blocks transferred.

Mode gives the type of error checking being used. It will be either Check (Checksum) or CRC (Cyclic Redundancy Check).

Rate gives the rate of transfer in bytes/second.

Files gives the number of files transferred in the current batch.

Retries gives the number of blocks that have been re-transmitted due to corruption. The downloader will automatically request re-transmission of a corrupted block, and will continue to do so until it is received uncorrupted, or until you select **Cancel** to abandon the download.

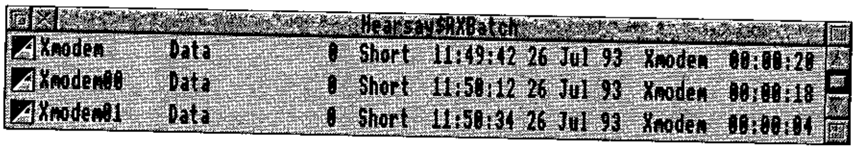
The file transfer status box disappears when the download is completed.

Default filenames

Hearsay saves received files using the filename which is transmitted by the host system along with the file. In the case of Xmodem, Xmodem 1K and ASCII transfers, no filename is transmitted, so Hearsay uses a default filename. The default name is the name of the protocol padded to 10 characters with a numeric suffix. So the default filenames for Xmodem files will be: `Xmodem`, `Xmodem0`, `Xmodem00`, `Xmodem000`, `Xmodem0000`, `Xmodem0001` etc. An option on the **Configure** menu allows you override this feature, and name each file as it is received.

RX batch

By default, downloaded files are saved in a batch directory within the Hearsay application. You can view these files using the **Show RX batch** option. The RX batch window lists all the files that have been downloaded, successfully or not. It's layout is similar to a standard directory display, but lists additional useful information about the transfer that has taken place.

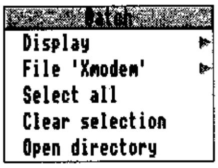


```
                          Hearsay$RXBatch
Xmodem      Data          0 Short 11:49:42 26 Jul 93 Xmodem 00:00:20
Xmodem00    Data          0 Short 11:50:12 26 Jul 93 Xmodem 00:00:18
Xmodem01    Data          0 Short 11:50:34 26 Jul 93 Xmodem 00:00:04
```

The words `Open`, `Short` and `Recvd` are placed after the file length to indicate whether the file was received correctly. `Open` indicates that the file is open i.e. it is still being received. `Short` indicates that the file received is not complete. `Recvd` indicates that the file was successfully received. In addition, the last two items in each entry show the type of file transfer protocol used, and the time taken to download the file.

You can run a received file by double-clicking on its icon in the batch window, but in most cases you will need to set the correct filetype using the **Set type** option on the batch sub-menu. Please refer to *Appendix D* for details of how to run archived files.

Many of the operations possible on the files in the batch are identical to those available in a standard directory display. Files may be copied to a new directory or disc by dragging in the usual way, or moved to a new location by dragging whilst holding down Shift. It is recommended that you regularly move files from the batch to another disc or directory, to ensure that the batch directory does not become full. You may select a group of files for copying or moving by clicking on them with Adjust.

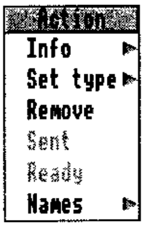


Click Menu over the RX batch window to display the batch menu.

Display

The **Display** sub-menu allows you to change the format of the batch display between **Large icons**, **Small icons** or **Full info**, and the sorted order of the files in the batch window.

File



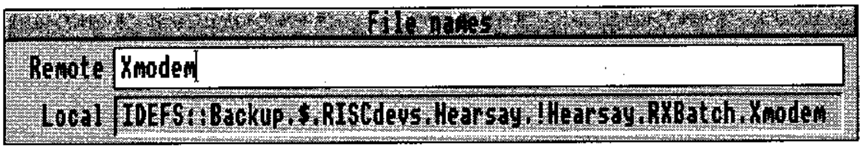
The **File** sub-menu allows you to apply an operation to the selected file, or the file under the pointer if there is nothing selected. If a number of files are selected, the operation will apply to all selected files.

Info gives detailed information about the selected file or files.

Set type allows you to set the filetype for the file.

Remove deletes the selected file or files from the batch, but the actual file itself is left unchanged on disc. To delete it, choose the **Open directory** option described below, and delete the file in the usual way.

Names displays the remote and local names for the selected file. The remote name is the name that was transmitted by the host system. The local name is the full pathname of the file saved, taking into account any changes to the filename. Hearsay may automatically rename certain files so their filename format is suitable for the local filing system (see **Use type alias** in the section on the **Configure** dialogue box).


File names

Remote Xmodem

Local IDEFS::Backup.$.RISCdevs.Hearsay.!Hearsay.RXBatch.Xmodem

Select all

This option selects all the files in the batch display.

Clear selection

This option de-selects all the files in the batch display.

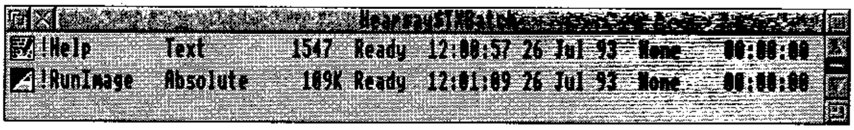
Open directory

This option opens the actual directory which contains the downloaded files. This directory is set by the system variable `Hearsay$RXBatch` in the `!Run` file, and by default is located within the Hearsay application directory. You can change the path of this directory using the **RX path** option on the **Configure** dialogue box.

Sending files

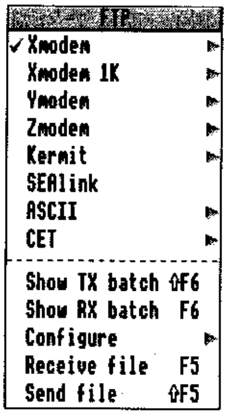
To send (or *upload*) a file or files, simply drag them into a terminal window. This operation does not make additional copies of the files, but places their names in a buffer. Files are not transmitted immediately, but placed in the TX batch, which is a buffer containing the names of all the files ready for transfer. The TX batch window is opened automatically by this operation, but you may open it at any time using the **Show TX batch** option.

If you hold down Shift when dragging files into the window, they will be replayed and not sent (see the chapter *Text terminals*).

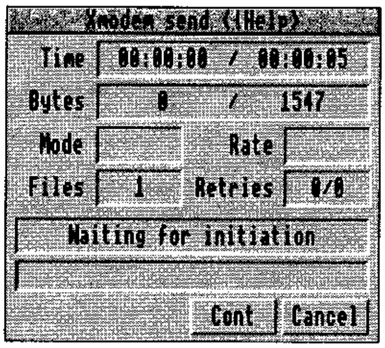


```
Hearsay$TXBatch
!Help       Text        1547 Ready  12:00:57 26 Jul 93  None    00:00:00
!RunImage   Absolute    109K Ready  12:01:09 26 Jul 93  None    00:00:00
```

The TX batch window lists all the files that are ready to be sent, or have been sent. It's layout is similar to a standard directory display, but it lists additional information about the status of the files. The word `Ready` after the file length indicates that the file has not be sent. The word `Sent` after the file length indicates that the file was successfully transferred. In addition, the last two items in each entry show the type of file transfer protocol used, and the time taken to download the file.



To send a file, choose a suitable file transfer protocol from the file transfer menu, and click on **Send file**. Hearsay will now transmit the file or files in the TX batch. Hearsay only transmits files marked with the word `Ready` i.e. those files which have not been transmitted. Any files that have previously been transmitted and indicated by the word `Sent`, are ignored. During the transfer the usual status box is displayed giving information about the progress of the transfer. You can stop the transfer at any time by clicking on **Cancel**.



Xmodem send (!Help)

Time 00:00:00 / 00:00:05

Bytes 0 / 1547

Mode

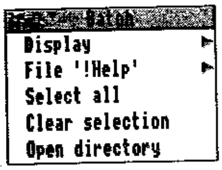
Rate

Files 1

Retries 0/0

Waiting for initiation

TX batch menu

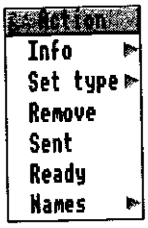


Click Menu over the TX batch window to display the batch menu.

Display

The **Display** sub-menu allows you to change the format of the batch display between **Large icons, Small icons** or **Full info**, and the sorted order of the files in the batch window.

File



The **File** sub-menu allows you to apply an operation to the selected file, or the file under the pointer if there is nothing selected. If a number of files are selected, the operation will apply to all selected files.

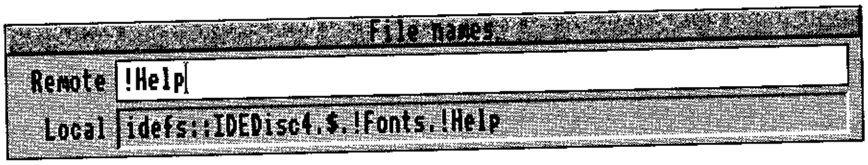
Info gives detailed information about the selected file or files.

Set type allows you to set the filetype of the selected file.

Remove deletes the selected file or files from the batch display only. The actual file itself is left unchanged on disc.

Sent and **Ready** indicate whether files have been transmitted, or are waiting to be sent. If a file has been transmitted, **Sent** will be ticked. When you use **Send file**, Hearsay only sends those files marked **Ready**, ignoring those marked **Sent**. These options may be used to change the status of files, allowing you to choose which files in a batch are to be transmitted. If you have a number of files in a batch ready to be transmitted, you can force Hearsay to ignore selected files by making them **Sent**. Likewise, if you have transmitted a file, and wish to send it again, you should make it **Ready**.

Names displays the remote and local names for the selected file. The local name is the full pathname of the file on the local filing system. The remote name is the name transmitted to the host system taking into account any changes to the filename. You may alter the remote name manually using this option. Hearsay may automatically rename certain files so their filename format is suitable for the other filing systems (see **Use type alias** in the section on the **Configure** dialogue box for further details).



Remote: !Help

Local: idefs::IDEDisc4.$.!Fonts.!Help

Select all

Selects all the files in the batch display.

Clear selection

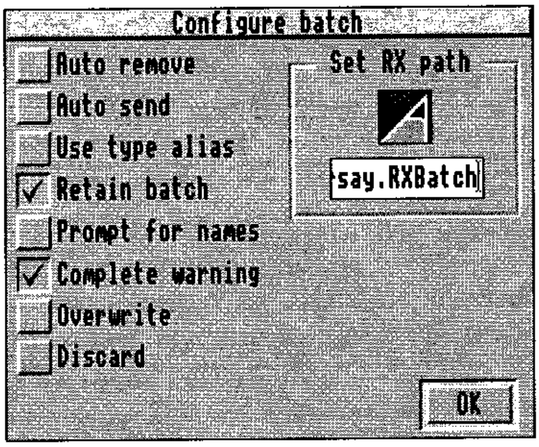
De-selects all the files in the batch display.

Open directory

This option opens a normal directory display on the actual directory used as a temporary store for files transmitted directly from RISC OS applications. For example if you prepare some text in Edit and drag directly from Edit to Hearsay, a temporary copy of the file is created, which is then transmitted in the normal way. As soon as the file has been successfully sent or if it is deleted from the batch using the **Remove** option, the temporary file is automatically deleted. This directory is determined by the system variable `Hearsay$TXBatch` set in the `!Run` file, and by default is located within the Hearsay application directory.

Configure batch

The **Configure** dialogue box allows you to alter the way file transfers occur.



Configure batch

□ Auto remove
□ Auto send
□ Use type alias
☑ Retain batch
□ Prompt for names
☑ Complete warning
□ Overwrite
□ Discard

Set RX path

say.RXBatch

If **Auto remove** is selected, once files have been successfully transmitted they are removed from the TX batch.

If **Auto send** is not selected (the default state), files dragged into a Hearsay terminal are simply queued in the TX batch until the **Send file** option is chosen. If **Auto send** is selected, files dragged into Hearsay are automatically sent using the currently selected file transfer protocol.

If **Use type alias** is selected when sending files, Acorn file types are mapped into MS-DOS extensions. So, for example, archives are given the extension `.arc`. On receiving files, MSDOS extensions are mapped into Acorn file types, so `file.zoo` is given filetype DDC. Type aliases are set using special script language functions; please refer to the script file `!Hearsay.AutoRun.!Custom`.

If **Retain batch** is selected, the contents of the RX and TX batch windows are retained between sessions.

If **Prompt for names** is not selected (the default state), received files are automatically saved without user intervention, in the default RX batch path. If it is selected, Hearsay will prompt with a standard save box every time a file is received. In this case, if a file is received you can name it as you wish, and drag it to the required destination. If a batch is received you can drag the batch icon from the save box, to the required destination directory for all the files in the batch.

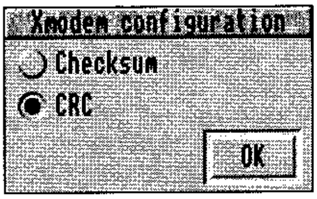
If **Complete warning** is selected, a warning beep will be made at the end of the download or upload.

If **Overwrite** is selected, existing files in the RX batch with the same name as downloaded files are automatically overwritten, otherwise a warning is given.

If **Discard** is selected, files only partially downloaded due to errors, are automatically discarded from the RX batch.

Xmodem

Probably the first true microcomputer FTP, was developed by Ward Christiensen in the late 70's called Xmodem. This has gone on to form the basis of numerous other protocols. Xmodem, is an eight bit protocol, so to use it you must be transmitting and receiving 8 data bits. It uses all codes between 0 and 255, so is incompatible with Xon/Xoff flow control. By sending files in packets, Xmodem can spot errors in transmission and fix them. Only one file at a time can be transferred using Xmodem, and files will have their length padded out to the nearest multiple of the packet length. The name of the file is not transmitted with the data, so a name must be provided by the receiver. Xmodem is a very popular file transfer protocol and you will find that it is available on most systems. However, If Ymodem, Zmodem or SEAlink are available on your host system you should use one of them in preference to Xmodem, since they are quicker, more reliable and transmit filenames.



Xmodem configuration

○ Checksum

● CRC

The block check may be set to **Checksum** or **CRC**. The original version of Xmodem, only used a checksum to detect errors in blocks. Later versions use a Cyclic Redundancy Check (CRC) which is more sensitive, and works in such a way that it can detect if the computer to which it is talking, supports this improvement. The default setting is **CRC**, which automatically detects whether to use CRC or checksum, and will work with most implementations. However, you may find one or two old Xmodem's which need you to use a checksum and will not cooperate with CRC mode. In these cases, you should set the block check mode to **Checksum**.

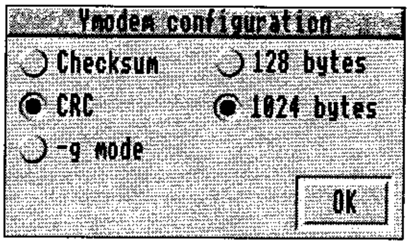
Xmodem 1K

This protocol was developed from the Xmodem protocol and uses 1024 byte packets. If there are very few errors encountered during file transfer, and if there is an appreciable time delay (e.g. in a satellite link or a packet switch network) it is possible to waste a significant amount of time while one computer waits for the other to signal that a file packet was received correctly so that it can send the next packet. By increasing the packet length, the total amount of time wasted in this way can be reduced.

Xmodem1K uses the same automatic method of changing between CRC and checksum block checks as Xmodem, so the **CRC** setting should suffice for almost all situations.

Ymodem

Ymodem is the batch extension of Xmodem 1k, and allows complete batches of files to be transmitted and received.



Ymodem configuration

○ Checksum ○ 128 bytes
● CRC ● 1024 bytes
○ -g mode

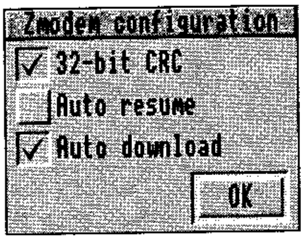
Ymodem make be configured to **CRC** and **Checksum** block checking just like Xmodem.

The option **-g mode** switches off error correction for received files. It is only useful on error-free or error-corrected communications links, but gives a useful improvement in performance.

The last two options on the menu allow you to configure the packet length. The default packet length for Ymodem is 1024 bytes. However, Ymodem senders can mix 128 byte blocks with 1024 byte blocks and the receiver will automatically adjust. So when receiving Ymodem files you do not need to worry about packet lengths, but when sending, you may choose the shorter length when errors on the line are a problem.

Zmodem

Zmodem is one of the latest developments in FTP's. Its special feature, is that the transmitter sends out a continuous stream of data, with checksums built-in. The receiver only acts when it sees an error. When this occurs, it tells the transmitter how much data it has, and transmission recommences from that point. Since the receiver does not normally send any data during a transfer, it is necessary to set the modem inactivity timeout to something larger than the transfer time, otherwise the modem will drop the line during the transfer. Zmodem is fast and reliable, and if it is available on your host system, should be used in preference to the other file transfer protocols.



Zmodem configuration

☑ 32-bit CRC

□ Auto resume

☑ Auto download

32-bit CRC enables improved error checking. It may only be used if the other end of the connection also supports it.

The **Auto resume** option allows you to resume a Zmodem download at any point in the file. Typically it is used to continue a download after it has terminated due to line problems or cancellation. Auto resume occurs when you download a file whose name matches a file in the RX batch. Instead of overwriting the existing file, the download continues from the end of the file.

If **Auto dload** is selected, Hearsay will automatically start a Zmodem download when it detects a Zmodem transfer starting on the host system. This means you can download files by simply telling the host system you wish to do so, with no further action required at your end.

SEAlink

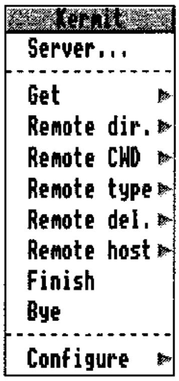
SEAlink is a development of Xmodem by System Enhancement Associates. The chief advantages over Xmodem, are that it is a batch protocol as well as a windowed protocol. No configuration options are provided. You will find that it is a quick and reliable protocol when communications take place over error free lines.

Kermit

Following Xmodem, but quite independently, the Kermit FTP was developed at Columbia University in the USA by Frank da Cruz and his team. The problem addressed, was transferring files between microcomputers and large mainframe machines using the somewhat restricted communications lines provided for terminal access. These are limited to 7 bits and are likely to use control codes for special purposes like Xon/Xoff. The result was Kermit, which can send 8 bit files over 7 bit lines, it preserves file lengths, sends filenames and can send any number of files at a time. Kermit is a batch protocol.

Kermit is probably the most complex of the FTP's to use. From the point of view of Hearsay, Kermit works most of the time like a standard FTP. To send or receive files, select the **Send file** and **Receive file** options on the FTP menu. Like the other batch protocols, you can select more than one file to send and will not be prompted for any file names when receiving files. File transfers then proceed as normal.

Server mode



The **Server** option puts Hearsay into Kermit server mode. To use Kermit server mode you would typically have two machines connected together. One machine would be put in server mode, and the other used to issue commands for transferring files. The Hearsay server recognises the following commands:

Get
Remote dir
Remote CWD
Remote type
Remote delete
Remote host
Finish
Bye

Server commands

You may issue server commands to another server using commands on the Kermit sub-menu. Many of them elicit a text response from the remote computer so you must have a text terminal window open. To use any of these commands, you must first put the remote computer into server mode by issuing the command `SERVER` (see below for further details).

Please note that only brief descriptions of the server commands are given below; for full details please refer to the user guide for the server concerned.

The **Get** command requests the Kermit server to send the file or group of files specified by the file specification entered in the writable menu box.

Remote dir. displays a directory of all the files that match the wildcarded file specification entered. If no file specification is given, all files in the current directory are listed.

Remote CWD means Change Working Directory. If no directory name is provided, the server will change to the default directory. Otherwise the server will attempt to change to the specified directory.

Remote type displays the contents of the specified file

Remote del. deletes the specified file or files.

Remote host passes the specified command to the server's command processor.

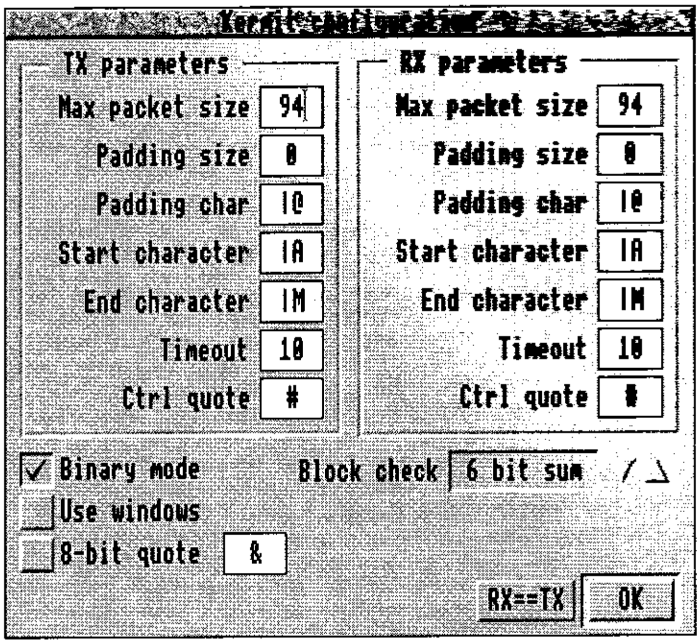
Finish shuts down the server and returns to Kermit command mode.

Bye shuts down and logs out the server.

It is important to note that not all Kermit implementations will support all server commands. If you attempt to use a server command that is not available, the error message `Unimplemented server command` will be displayed. Please note also that servers vary in the exact way they execute commands, so if in doubt consult the user guide for the server concerned.

Kermit configuration

Kermit has a large number of configuration options. but in most circumstances the default settings are suitable.



Kermit configuration

TX parameters		RX parameters	
Max packet size	94	Max packet size	94
Padding size	0	Padding size	0
Padding char	|@	Padding char	|@
Start character	|A	Start character	|A
End character	|M	End character	|M
Timeout	10	Timeout	10
Ctrl quote	#	Ctrl quote	#

☑ Binary mode Block check 6 bit sum

☐ Use windows

☐ 8-bit quote &

TX parameters

Max packet size is the maximum allowable packet length. Hearsay will use long packet mode if a packet length of over 94 is specified. The maximum length is 2048, but a recommended length is 1024.

Padding size is the number of pad characters sent before each block.

Padding char defines the pad character code.

Start character defines the character used to indicate the start of a packet.

End character defines the character used to indicate the end of a packet.

Timeout is the number of seconds before the transmitter will re-send a packet.

Ctrl quote is the character used to indicate that the following character is a control character. Kermit translates codes 0 to 31 into codes 32 to 63 and prefixes them with this character.

RX parameters

RX parameters are the same as those described above, but for received data only.

If **Binary mode** is not selected you should only transfer ASCII files. Binary mode should be selected if you wish to transfer binary files.

If **Use windows** is selected, Kermit will use sliding windows for file transfer. Windowed Kermit will improve the performance of the file transfer, but can only be used if both sender and receiver support it.

8-bit quote is used to enable 8th bit quoting, and to specify the character used to prefix characters in the range 128 to 255. 8th bit quoting allows Kermit to transfer 8-bit characters along 7-bit channels.

Block check allows you to choose the type of block checking to be used when error checking packets. It may be set to to **1 6-bit sum**, **2 12-bit sum**, or **3 16-bit CRC**.

RX==TX makes the RX parameters the same as the TX parameters.

Using Kermit

Many people who can happily transfer files using Xmodem have some difficulty with Kermit. To help, we present here a typical Kermit session on an imaginary database. Output from the remote computer is shown in feint courier e.g. `remote output`, and user input in bold courier e.g. **`user input`**.

```
Welcome to the Kermit Database

Please Logon: USER

Password: PASSWORD

Logon to the Kermit Database, Thur 5th of March
1991.
```

(We have logged onto the host, and now wish to use Kermit to transfer files, so we type Kermit)

```
>Kermit

Kermit now running

Kermit>
```

(Kermit is now running on the remote computer, shown by the Kermit prompt. We are now in Kermit command mode on the remote machine and we can issue commands to it. We issue the command ? to get help.)

```
Kermit>?
Available commands are: Send, Receive, Set,
Show, Server, Finish...
```

(Suppose we want to receive a file PROGRAM from the remote host, we type:)

```
Kermit>SEND PROGRAM
```

(At this point we select **Receive file** on the **File transfer** menu and the file would be transferred. Next we may decide to send a file called MYPROGRAM to the remote host.)

```
Kermit>RECEIVE MYPROGRAM
```

(Having told the remote computer that we are going to send it a file which it should receive, we would go to the **File transfer** menu and send the file in the normal way. Such 'file at a time' transfers work well, but it would be better to make use of the Kermit server function.)

```
Kermit>SERVER
Entering server mode. Escape to command mode on
your machine.
```

(We can now use any of the Kermit server commands. For instance `Get` to transfer files from the remote computer or maybe `Remote dir.` to see which files are available. It is also possible to use the normal **Send file** option to transmit files to the server; however, **Receive file** cannot be used in this mode. Finally, to terminate our session, we might select `Finish` which would return us to Kermit command mode on the remote computer from which we could exit to the usual user mode by giving the `FINISH` command. However, instead we select `Bye` which exits from the server and logs you out.)

```
Bye from the Kermit Database.
Please call again.
Logged off 22:00:45; Connect Time 02:34:23.
```

ASCII

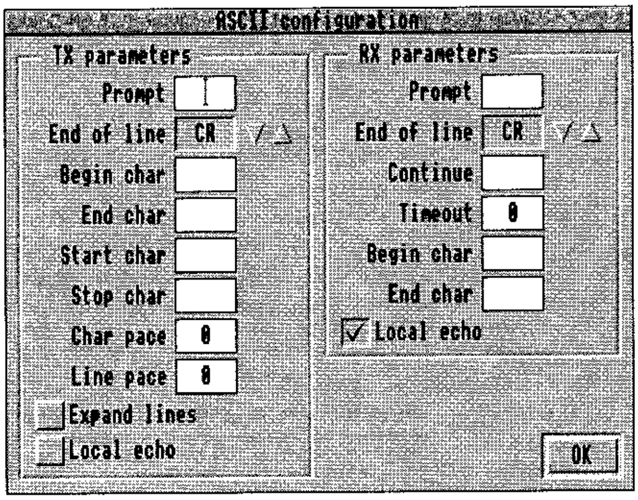
Hearsay comes with an extensive ASCII file transfer facility. ASCII file transfer has many disadvantages, (e.g. lack of error correction), and generally it should be avoided if at all possible. The one situation in which you are likely to need it is when you are trying to transfer files to a computer with no file transfer facilities at all. Here you will have to transfer files by making your computer 'type' the file to the remote computer in the same way as you might type a program in from the keyboard. Similarly, when receiving a file you will have to make it list the file for you, and then capture the output to disc. The extensive spooling facilities built into the text terminals are used to do the latter. ASCII file transfer does all this automatically. It should normally only be used with 7 bit text files, although if both ends of the link allow it, 8 bit binary files can also be transferred.

The **Cont** icon in the file transfer status box has a special use in the ASCII FTP. There are many situations where the ASCII FTP is waiting for a particular byte to arrive from the remote computer before continuing. If this byte fails to arrive, then both computers will wait forever for each other to do something. Selecting **Cont** breaks this deadlock and continues the transfer as if the desired byte had arrived.

To perform an immediate ASCII upload, hold down Ctrl and drag a text file into a terminal window.

ASCII Configuration

The ASCII file transfer dialogue box allows you to configure the ASCII transmit and receive parameters.



ASCII configuration

TX parameters
Prompt
End of line: CR
Begin char
End char
Start char
Stop char
Char pace: 0
Line pace: 0
□ Expand lines
□ Local echo

RX parameters
Prompt
End of line: CR
Continue
Timeout: 0
Begin char
End char
☑ Local echo

Do not attempt to configure the ASCII transfer unless you are sure the values you are entering are correct. It is all too easy, for example, to enter a prompt character and then find the transfer keeps hanging, because the remote host does not send prompts.

TX parameters

Prompt is the character the remote computer will send when it wants the next line to be sent. The **End of line** character determines when to wait for the prompt character. Typically, the remote computer may send $ or > when it expects you to send it a line. You should ensure this writable menu box is empty if you want lines to be sent continuously.

End of line is the character you want sent whenever an end of line is encountered in the file being sent. Hearsay works out itself, what is an EOL in the file being sent, and replaces it with the sequence specified. **End of line** may be set to **CR**, **LF**, **CRLF**, **LFCR** or **Raw**. If **Raw** is selected, the end of line marker will be passed on unaltered. Use this setting if you want to send a file as it exists on disc.

Begin char is the character added to the start of any file sent. It is the character the remote computer needs to tell it to start a transfer.

End char is the character added to the end of any file sent. It is the character the remote computer needs to tell it the transfer is complete.

Start char is the character that the remote computer will send to tell you to restart sending after it has sent you a stop character.

Stop char is the character that the remote computer will send to tell you to pause sending temporarily.

Char pace is the delay introduced between each transmitted character in 1/100ths of a second. You may wish to slow down transfers in this way, because some systems cannot accept data to be typed in at any great speed; being optimised for human typists. If you send them data too quickly, you may find bits of your file are missing.

Line pace is the delay introduced between each transmitted line in 1/10ths of a second. The reasons for doing this are much the same as for introducing a character pace.

If **Expand lines** is selected Hearsay converts CR CR to CR `space` CR. This useful when transmitting ASCII files to host systems which terminate their input with two consecutive carriage returns.

Local echo controls whether transmitted data is echoed in the terminal.

RX parameters

Prompt is the character that will be sent after each **End of line** character is received. Some systems may require this to be set to the Xon character |Q.

End of line is the character marking the end of a line. It is the character after which the prompt for the next line will be sent, and may be set to **CR** or **LF**. Received data is just spooled to the file with no translation.

Continue is the character sent after the time (in 1/10ths of a second) specified by **Timeout** has expired. It is useful in situations where the computer sending the file, sends a few lines and then pauses until it receives a given character. You can enter here the character you want to use to nudge the remote computer into action.

Begin char is the character that the remote computer will use to signify the start of a transfer.

End char is the character that the remote computer will use to signify the end of a transfer.

Local echo controls whether transmitted data is echoed in the terminal window.

CET

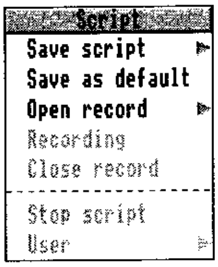
CET is a completely separate FTP, and is used almost exclusively on Viewdata systems. Again, this is a protocol which has developed to allow error free file transfer within the constraints of Viewdata systems. To briefly put it in the same context as the above FTP's, it allows 8 bit file transmission over 7 bit data lines, and preserves filenames and lengths.

Further details of CET telesoftware downloading and configuration are given in the chapter *Viewdata terminal.*

7 Script menu

The **Script** menu on the main menu gives various options concerning script language files.

Save script



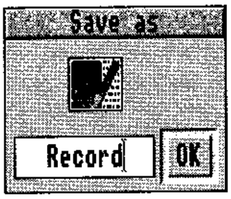
This option saves the current configuration of Hearsay in a script file with the name specified. The script file generated contains a set of functions which configure every aspect of Hearsay including all terminal settings. The script can be re-loaded by double-clicking on it's icon, or dragging it to the Hearsay icon on the icon bar. Alternatively, you can attach script files to entries in the number directory so that they are executed before and after dialling numbers from the directory. Please refer to the chapter *CScript* for further details.

Save as default

This option saves the script file `!Hearsay.AutoRun.!Config`. Script files in the `AutoRun` directory are automatically executed when Hearsay is run, so this option allows you to customise the Hearsay start-up configuration.

Open record

Open record allows you to open a script file into which key presses to the terminal, prompts from the host, and pauses are recorded. Typically it may be used for recording logon dialogues between your terminal and the host, which may be replayed at any time by running the script file. To record a logon dialogue, first start recording by dragging the script icon to a suitable directory in the usual way.



Now dial the system you require and logon in the usual way. Once you have logged-on, close the recording using the **Close record** option. The script file produced is a template which may be fine tuned in Edit or another text editor.

To test the recording, logoff the system and disconnect, then re-dial it again. As soon as your modem connects, double-click on the script and it should automatically logon for you. Typically you may use this script file as a logon script in the telephone directory to perform auto logon. The format of the script is a `main()` function which is executed when the script is run, followed by one or more of the following standard script functions:

`pause(time)`	causes a delay of `time` centi-secs
`kprints(s)`	sends string `s`, as though typed
`getprompt(prompt, time)`	waits `time` cent-seconds for `prompt` to be received

Recording

This option is used to switch recording on and off. A tick next to the option indicates that recording is on. Note that this option does not open and close the record file, but simply switches the stream on and off.

Close record

This option stops recording, and closes the script file.

Stop script

Stop script stops the script file that is currently running. It can only be used if the script file regularly polls the WIMP using the `pause()` function. If this is not the case, you can stop script files using Ctrl-Esc.

User

This menu lists the defined macros whose names commence with `M_`. Clicking on a name will cause the macro to be expanded. Typically the macro body will be of the form `{program}`, and hence cause a script to be run. However, the body could be just a string, in which case it would be sent, just as if it has been typed. The prefix `M_` is not shown on the menu. See the chapter *Macros* for details of defining macros.

The **User** menu may be displayed in place of the main Hearsay menu by pressing Ctrl-Menu. This operation may be reversed using the script function `setmenusense(REVERSE)`, causing Menu to display the User menu, and Ctrl-Menu to display the usual Hearsay main menu. This feature allows you to customise Hearsay for specific applications, by providing a simplified main menu of options. See the chapter *CScript* for details of the function `setmenusense()`.

8 Text terminals

This chapter describes the operation of the scrolling text terminals in Hearsay. These terminals are the VT320, VT102, VT52, ANSI, Teletype and the text terminals in Tektronix and Campus 2000. Each terminal emulates a different set of escape sequences, but the general operation is identical.

Scrolling text terminals normally display 24 lines of 80 characters, which scrolls when the screen is full. The display can be changed to 132 columns and have a 25th status line; these can be set from menus or changed automatically by the host system.

Hearsay uses a slightly non-standard window for the scrolling text terminals to enable it to display the full 80 columns used by true VT and ANSI terminals. The window toggle and resize icons are positioned on the title and horizontal scroll bars respectively instead of the vertical scroll bar. With this arrangement the window can display a full 80-columns with the vertical scroll bar pushed off the right edge of the screen. Click on the toggle icon to reduce the window with so that the vertical scroll bar is visible. The vertical scroll bar may then used to scroll through the capture buffer.

Screen modes and colours

The scrolling text terminals may be used in any screen mode, but we recommend that you use mode 12, or modes 20 or 27 if you have a multi-scan monitor. The ANSI terminal is a colour terminal and uses the standard set of ANSI colours. Unfortunately, these colours are not all present in the standard Archimedes 16 colour palette, so Hearsay uses the best equivalents available. If you want to use the true ANSI palette, either change to a 256-colour screen mode (which uses more memory and will slow your system down), or load `!ANSIpal` from the Hearsay Extras disc.

Please note that loading `!ANSIpal` will change the palette for all applications running on your computer.

Mouse operation

Text terminals respond to commands in the form of words or characters typed at the keyboard. Normally they are not acted upon until Return is pressed, so mistakes can be corrected using Delete or Backspace (depending on the host system). In addition, the mouse may be used as follows:

single-click Select	send character under pointer
single-click Adjust	send carriage return
double-click Select	home cursor to pointer (only in Application cursor mode)
double-click Adjust	select from cursor to pointer

Therefore the mouse can be used to select from an on-screen menu, by moving the pointer over the menu item number or letter and clicking Select. If a carriage return is needed, just click Adjust. Hearsay allows blocks of text to be selected for saving, printing, transmitting etc. To make a selection click Select at the start of the selection and drag to the end in the normal way. When a selection is highlighted, the mouse buttons do the following:

single-click Select	clear the selection
single-click Adjust	extend/reduce the selected block

Please note that while there is text selected in the terminal, all input into the terminal is suspended. When the selection is cleared, terminal input is resumed.

Function, keypad and cursor keys

The function keys F1 to F11 generate a number of useful functions when pressed on their own or with Shift. Please refer to *Appendix E* for a summary of these functions. When pressed together with Ctrl or Action (the right-hand Ctrl key) they may generate user-defined strings. In fact virtually all the keys on the keyboard may be re-defined using the macros facility described in the chapter *Macros*. If Action is pressed with a function key in the VT320/VT100 terminal, a standard escape sequence is transmitted (see *Appendix B* for a list of these).

The keys on the numeric keypad generate normal characters in the Teletype terminal, but in other terminals generate special escape sequences in application keypad mode. These escape sequences are detailed in *Appendix B* and *Appendix C*.

In the Teletype terminal the cursor keys generate cursor movement, but in the other terminals they generate escape sequences. In application cursor mode, these escape sequences cause cursor movement.

Application cursor and keypad modes may be chosen from the **Keyboard** dialogue box, but are normally set automatically by the host system if required.

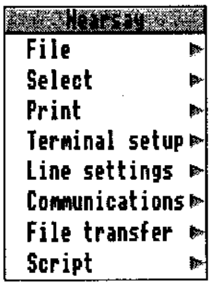
Capture buffer

Hearsay stores all received data in a capture buffer. The normal terminal window displays the last 24 lines of this buffer. The capture buffer allows you to scroll back through text that has disappeared off the top of the screen, until you reach the start of the buffer. By default, Hearsay has a capture buffer of 48 rows i.e. 2 screens of text, but this may be extended if memory permits (using an option on the **Misc** dialogue box). Once the capture buffer is full, text at the start of the buffer is lost as the buffer scrolls to accept new data. You may scroll through the capture buffer using the vertical scroll bar, or the key presses given below.

Page Up	scroll back through capture buffer
Page Down	scroll forward through capture buffer
Shift Page Up	scroll back one page
Shift Page Down	scroll forward one page
Ctrl Page Up	go to start of capture buffer
Ctrl Page Down	go to end of capture buffer
Copy/End	return to cursor position

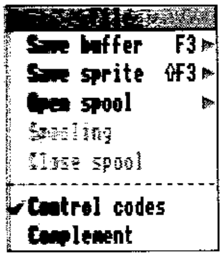
Remember that you may need to resize the window to display the vertical scroll bar. To do this click on the window toggle icon.

Menu options



The options available in the scrolling text terminals are chosen from the sub-menus **File**, **Select**, **Print** and **Terminal setup**. These menus are chosen from the main Hearsay menu, which is displayed by clicking Menu over the terminal window.

File menu



The **File** menu provides a number of options for saving data from the terminal in different formats.

Save buffer

This option saves the entire contents of the capture buffer. The format of the data saved is controlled by the **Control codes** option.

Save sprite

Save sprite saves the terminal contents as a sprite file.

Open spool

This option spools all data to file as it is received. There may only be one spool file open in Hearsay at any time, and that file is common to all the terminals. The format of the data spooled is controlled by the **Control codes** option described below.

Spooling

Spooling switches spooling on and off, and is shaded until a spool file is opened. When you first open a spool file, this option will be ticked indicating that spooling is on. Click on **Spooling** to toggle spooling on and off.

Close spool

This option switches off spooling and closes the spool file.

Control codes

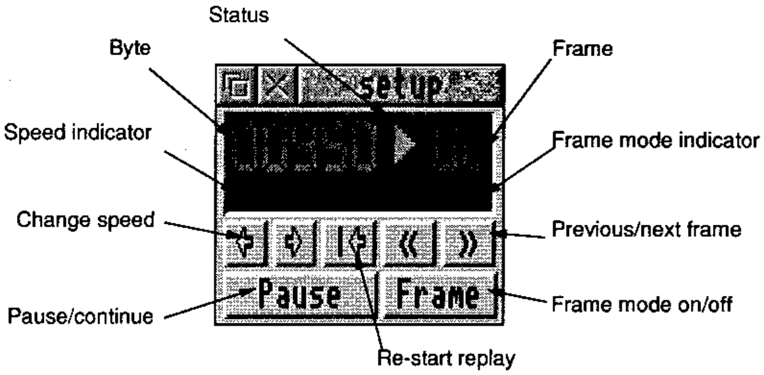
If **Control codes** is selected, the **Save buffer** and **Open spool** options create a data file and save both plain text and terminal control codes and escape sequences. If **Control codes** is not selected, these options create a text file and save plain ASCII text only.

Complement

If **Complement** is selected, the **Save sprite** option inverts the sprite before saving. This may be useful if you eventually want to print the sprite, because it changes the black terminal background colour to white which will reduce ribbon wear on dot-matrix printers.

Replaying spooled files

If Shift is pressed and a spooled file is dragged into a terminal, that file is replayed in the terminal. If the spooled file includes control codes, the replay session will include colours and style effects from the original session, otherwise only plain text will be replayed.



During replay, a dialogue box is displayed which allows you to control how the file is replayed. To stop the replay, close the dialogue box.

Byte shows the byte currently being replayed.

Speed indicator shows the speed of the replay. The greater the number of blocks, the faster the replay.

Change speed allows you to increase or decrease the speed of replay.

Pause/continue allows you to pause the replay. When paused, this icon changes to **Continue** which allows the replay to resume.

Re-start replay will start the replay again from the beginning.

Status gives a visual indication of the status of the replay. It shows whether the replay is waiting at a frame, paused or completed.

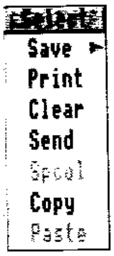
Frame shows the number of the frame currently being replayed.

FRA is displayed if frame mode is on.

Previous/next frame allow you to step to previous or next frames i.e. to previous or next 'clear screen' character.

Frame mode on/off switches frame mode on and off. If frame mode is on, the replay will pause at every 'clear screen' character giving you chance to read the previous data before the screen clears. To continue the replay, click on **Continue**.

Select menu



The **Select** menu provides options for manipulating selected text. Details of how to select text are given on page 62.

Save

Save allows you to save the selected text. The format of the data saved is controlled by the **Control codes** option on the **File** menu.

Print

This option prints the selected text. It is printed to the printer type specified on the **Choices** dialogue box. Please refer to the chapter *Icon bar menu* for further details.

Clear

This option clears the selection. You can also clear the selection by clicking Select in the terminal window.

Send

Send transmits the selected text as though it had been typed. This option is useful if you wish to send anything that is currently displayed in your terminal. For example, if you need to re-send a password, just select the required text and choose **Send**.

Spool

Spool appends the selected text to the spool file. A spool file must be open, but spooling does not need to be enabled. The format of the text spooled is controlled by the **Control codes** option on the **File** menu. This option will be shaded if there is no spool file open.

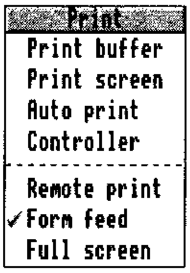
Copy

Copy makes a copy of the selected text in memory, where it is retained until required. The copied text can be recalled and transmitted using the **Paste** option described below. Copied text is lost when the terminal is closed, when **Min memory** is used, or when you quit Hearsay.

Paste

Paste transmits the text copied by the **Copy** option described above.

Print menu



The **Print** menu controls printing from the scrolling text terminals. Text is printed to the printer type specified on the **Printer** sub-menu on the **Choices** dialogue box. This can be via a RISC OS printer driver, or directly to your printer port. If you use a RISC OS printer driver, the output is not printed immediately, but buffered. The buffer is not printed until a full page is received, or until you force the buffer to be printed, using the **Min memory** option on the icon bar menu. The size of the printed page is defined using `setprinter()` in the `!Custom` auto-run script. Please refer to the chapter *CScript* for more details.

Print buffer

This option prints the entire capture buffer

Print screen

Prints the current screen, i.e. the last 24 lines of the capture buffer.

Auto print

If **Auto print** is selected, all data received by the terminal is printed. If **Remote print** is selected, this option may controlled by the host.

Controller

If **Controller** is selected, all data received is echoed to the printer, but not to the terminal. It may be used by the host system to send special escape sequences to the printer, which could possibly upset the terminal. If **Remote print** is selected, this option may controlled by the host.

Remote print

If **Remote print** is selected, Hearsay will act upon escape sequences received which control the printer. It should be selected if you want the host system to control your printer. If **Remote print** is not selected your printer cannot be switched on by the host system or by corrupted data.

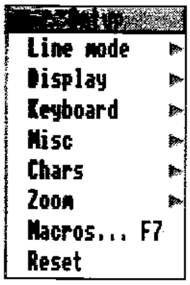
Form feed

If **Form feed** is selected, a form feed is printed after each print operation.

Full screen

If **Full screen** is selected, the **Print screen** operation will print the entire screen. If it is not selected, only the scrolling region is printed.

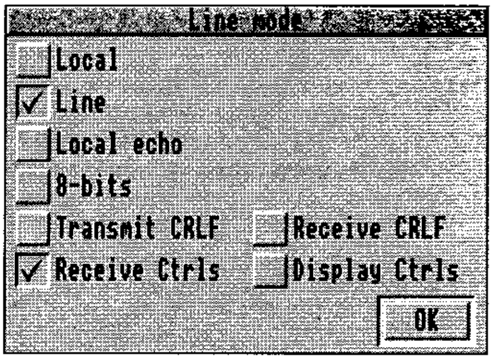
Terminal setup menu



The **Terminal setup** menu allows the current terminal to be configured.

Line mode

The **Line mode** dialogue box contains options concerned with the communications line, echo and filters.


Line mode

□ Local

☑ Line

□ Local echo

□ 8-bits

□ Transmit CRLF □ Receive CRLF

☑ Receive Ctrls □ Display Ctrls

If **Local** is selected, terminal output is disabled and echoed back into the terminal. You may wish to use this option to test features of the terminal while you are offline.

Line controls terminal input. If **Line** is not selected, the terminal does not read from the serial port. In normal use, **Line** should be selected.

If **Local echo** is selected, all output from the terminal is echoed in the terminal window. Normally this feature is not required since the host computers usually echo any received characters back. However, you may need this option if you are connected to a simple host which does not echo received characters.

8-bits determines whether the terminal recognises 7 or 8 bit commands. This option should normally be set if you are using the VT320 terminal.

If **Transmit CRLF** is selected, all carriage return characters transmitted, are sent as two characters; CR LF.

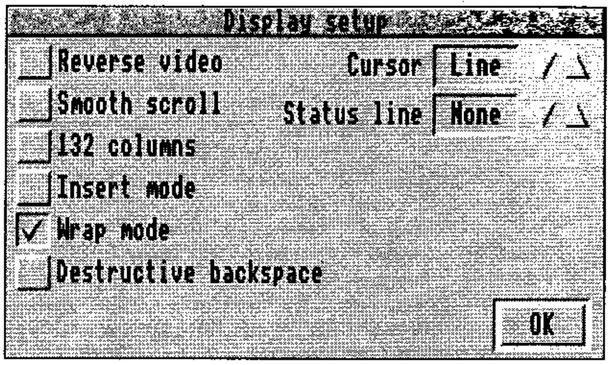
If **Receive CRLF** is selected, all carriage return characters received, are interpreted as two characters; CR LF.

Receive Ctrls causes the terminal to act upon control codes received.

Display Ctrls causes the terminal to display control codes using the standard escape sequences (listed in the chapter *Macros*), and not act upon them. Control codes are non-printable ASCII characters in the range 0-31 and 127- 255.

Display

The **Display** dialogue box contains options to control the text terminal display.



Display setup

☐ Reverse video

☐ Smooth scroll

☐ 132 columns

☐ Insert mode

☑ Wrap mode

☐ Destructive backspace

Cursor: Line

Status line: None

Reverse video reverses the foreground and background colours for the terminal displays. This normally gives black text on a white background.

Smooth scroll causes the terminal to scroll in small steps rather than big jumps.

132 columns switches the terminal to a 132-column operation. In 80-column modes you will have to scroll the terminal window to access the extra columns. In wider screen modes, such as mode 16, all columns will be visible. This option also clears the terminal window and buffer.

If **Insert** is selected, characters received are inserted at the cursor position, with the rest of the line shifted right to make space. If **Insert** is not selected, characters received overwrite any existing text.

Wrap controls *Auto wrap* and determines what happens when text is written beyond the 80th (or 132nd) column. If it is selected, characters appear at the start of the next line, otherwise they just overwrite the 80th character.

Destructive backspace determines what happens when the terminal receives the backspace character. If this option is selected, Backspace will delete the character to the left of the cursor, otherwise it will just move the cursor one character to the left.

Cursor allows you to choose the cursor type. The sub-menu allows you to choose **None, Line** or **Block**.

Status line displays a menu with further options to control the 25th line of the terminal display.

If **None** is selected, the status line will not be displayed.

If **Local** is selected, a local status line will be displayed, giving the LED status, baud rate, word format, time on-line (or units used), and the cursor coordinates.

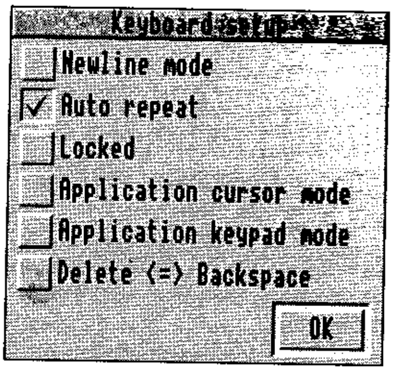
If **Host** is selected, the terminal will switch to a 25 line display, but the contents of the 25th line will be under control of the host.

If **Tabs** is selected, the position of the VT tabs will be shown on a ruler on the 25th line so that they may be edited.

click Select	add tab at the pointer position
click Adjust	remove tab at the pointer position
Ctrl Select	clear all tabs
Shift Select	set tabs at every 8 characters

Keyboard

The **Keyboard** dialogue box provides options concerning keyboard operation.



□ Newline mode
☑ Auto repeat
□ Locked
□ Application cursor mode
□ Application keypad mode
□ Delete (=) Backspace

If **Newline mode** is selected, carriage return is transmitted as CR LF. In addition, line feed characters received are converted to LF CR. Whether this mode is appropriate for you, depends on the host system. If you get no line feeds, **Newline mode** should be enabled, whereas if you get double line feeds, it should be disabled.

If **Auto repeat** is selected, the keyboard will *auto repeat*. The repeat rate can be configured using the RISC OS configure commands `*Configure Delay` and `*Configure Repeat`.

If **Locked** is selected, the keyboard is disabled. This option is intended for use under direct control of the host system.

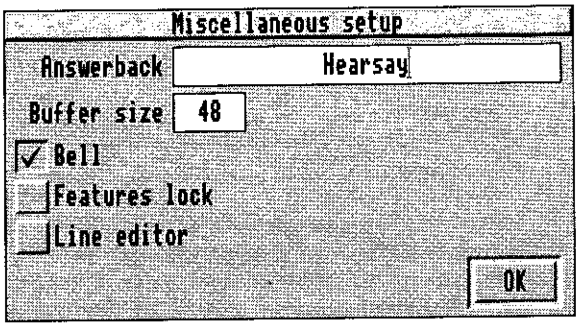
If **Application cursor mode** is selected, the cursor keys generate escape sequences in the VT terminals which cause cursor movement. If this option is not selected, the cursor keys generate special sequences which are interpreted and acted upon by the host system.

If **Application keypad mode** is selected, the keypad keys generate special escape sequences in the VT terminals which are interpreted and acted upon by the host system. If this option is not selected, the keypad generates the same characters as those shown by the legends on the keys (except the top row, which always produce special codes).

Delete<=>Backspace swaps the operation of the Delete and Backspace keys.

Misc

This dialogue box contains a number of miscellaneous options concerned with the scrolling text terminals.



Answerback Hearsay

Buffer size 48

☑ Bell

□ Features lock

□ Line editor

Answerback allows you to define a string which identifies your terminal. The host system may request this string.

Buffer size allows you to specify the size of the capture buffer as a number of lines. The default size is 48 lines, which is equivalent to two terminals screens. You may enter any size from 25 lines upwards. Please note that each 80-column line uses 320 bytes, so entering large values will take up valuable memory on machines with smaller amounts of memory. We recommend that on a 4Mb machine, the capture buffer is set to 256 lines. Further details about using the capture buffer are given at the start of this chapter.

If **Features lock** is selected, terminal options **Smooth scroll, Reverse video, Auto repeat, Locked** and **Tabs** are locked and cannot be changed by the host system.

If **Bell** is selected, Hearsay will sound a beep when it receives the ASCII code 7, otherwise this code is ignored.

Line editor opens a single line editor which may be used to enter and edit text before it is transmitted to the host. How useful it is depends on the host system. If the host only requires single key to be pressed, or if it has its own line editor, then the line editor will not be of much use. However, if you need to type commands lines to the host, then it may prove useful for correcting typing mistakes before the text is transmitted. After selecting this option, a single line window will appear towards the bottom of the text terminal.

Into this you may type text, which may be edited using the cursor keys, Delete and Copy in the normal way. As usual you may delete the entire line using Ctrl U. Once you are happy with the text, press Return to transmit it to the host.

The line editor is buffered, so you can use the cursor up and down keys to step to other lines of the editor. Pressing Return will transmit the line shown. The buffer is 32 lines long, and if you step past line 32 you will return to the first line.

The line editor directs all output through the macro processor, so you may enter key macros, user macros and even script language statements by enclosing them in braces. It is particularly useful for debugging script programs.

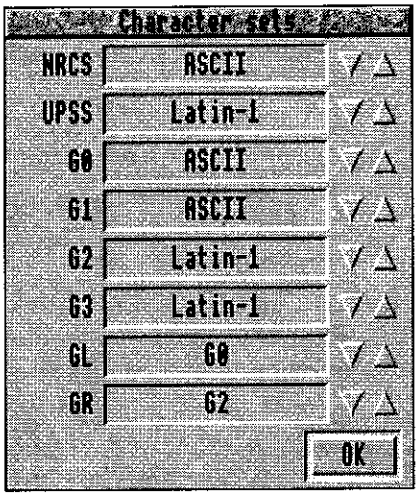
For example the following prints in the terminal the value of a variable which is present in the script file currently running:

```
{int variable = 1; tprints(variable);}
```

Notice that this may be a compound statement, so you can declare variables, put in loops etc.

Characters

This dialogue box allows you to select the character sets used by the VT terminals. Normally you do not need to alter the character sets, as they are controlled directly by the host system.



NRCS	ASCII
UPSS	Latin-1
G0	ASCII
G1	ASCII
G2	Latin-1
G3	Latin-1
GL	G0
GR	G2

NRCS is the National Replacement Character Set. The host system can automatically switch to the NRCS.

UPSS is the User Preferred Supplemental Set. The host system can issue a request to find out what UPSS is set to.

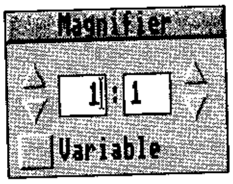
G0 to **G3** can each be set to one of the character sets on the menu. The host system may switch any of G0 to G3 into the top or bottom half of the current set.

GL specifies which of the sets G0 to G3 is shifted into the bottom half of the current set i.e. characters 32 to 127.

GR specifies which of the sets G0 to G3 is shifted into the top half of the current set i.e. characters 160 to 255.

Zoom

Zoom opens the standard magnifier dialogue box, and is used for scaling the text terminal.



The magnification is expressed as a ratio; for example 2:1 means twice normal size, 3:4 means three-quarters normal size, and so on. You can change the magnification ratio by clicking on the arrows, or by typing in a new value. To remove the magnifier dialogue box, press Escape or move to a different menu item. If **Variable** is selected, the terminal is scaled to the size of the window.

Macros

Macros opens a dialogue box which allows you to re-define virtually all the keys on the keyboard, and to define macros that may be used in those definitions. Full information is given in the chapter *Macros*.

Reset

Reset resets the terminal. It clears the screen and homes the cursor, but does not reset any of the terminal configuration settings or display attributes.

9 Viewdata terminal

This chapter describes the operation of the Viewdata Terminal in detail. It assumes that you are online to a Viewdata service such as Prestel, and have entered your ID and password. Please refer to the chapter *Getting started* for simple instructions on how to call and logon to the service you require. The instructions given in this chapter assume that you are connected to Prestel, but most other Viewdata services work in much the same way. Prestel specific functions are identified in the text.

The Viewdata terminal is displayed in a window comprising 24 rows of 40 columns. It may be used in any screen mode, but we recommend that you use mode 12, or mode 20 if you have a multi-scan monitor. Unfortunately, all Viewdata colours are not present in the standard Archimedes 16 colour palette, so Hearsay uses the best equivalents available. If you want to use the true palette, either change to a 256-colour screen mode, or load `!ANSIpal` from the Hearsay extras disc. There are disadvantages with both these options; 256-colour modes are slower and use more memory than 16-colour modes, and using `!ANSIpal` will change the palette for all applications running on your computer.

Minitel terminal

Please note that much of what is described in this chapter also applies to the Minitel terminal. The major differences are described below:

1. The keypad is in French (see `Scripts.ReadMe` on the Extras disc for details of how to generate an English keypad and how to set the keypad to send original codes or newer X3 codes).
2. The Viewdata editor is not available.
3. Frame send and Mailbox functions are not suitable for Minitel.
4. Frames are saved using the CEPT 2 file format.
5. Hearsay supports the extra 25th row in Minitel. This is positioned off the top of the terminal window, but may be scrolled into view if required.
6. Minitel frames may be tagged, but you cannot mix Viewdata and Minitel frames in one buffer.

Moving around the database

Viewdata databases often consist of many thousands of pages of information. This section describes how to access that information both effectively and economically.

Route number keying

It is possible to browse through a Viewdata database using only the digits 0 to 9, * and #, following the route indicated on each page. Usually only single digit numbers are needed, but sometimes you may need to key a 2 or even 3 digit number. In this case each digit takes you to an index page that leads to the next number. If you press the keys quickly, you won't see the the intermediate pages at all.

A Viewdata page can continue over 26 frames identified by the letters a to z after the page number. These frames cannot be viewed out of sequence. The # key takes you forward a frame or page, and for convenience, the Return key sends #. A true carriage return is sent using Shift-Return.

Going direct to pages

Although route number keying is quick to learn and easy to use, it is slow and uneconomical. If you know the number of the page that you want, you can go directly to that page at any time by typing * followed by the page number and then #. For example, to go to page 5000, type: `*5000#`.

In addition to page numbers, many pages in Prestel are assigned a unique keyword. You can go to these pages at any time by typing * followed by the appropriate keyword and then #. For example, to go to the Prestel mailbox front page, type: `*MAILBOX#`.

You can often find page numbers and keywords by looking in the database directory, or by making a note of them as you browse through the database.

On-screen routeing

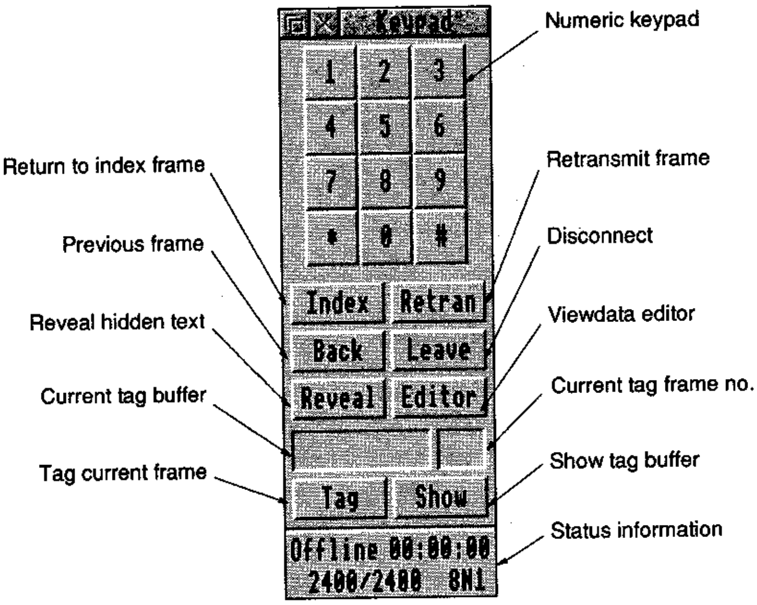
The Viewdata terminal in Hearsay may be entirely mouse-driven. Simply move the pointer over the route number you require, and click Select. The number may be in the terminal window or on the numeric keypad, and may be one or more digits long. For convenience, the Adjust button is equivalent to the Return key i.e. #.

You can also click on a page number or keyword that appears in the terminal, to go directly to that page, provided that the page number or keyword is preceded with a *, and ends with a #. Prestel often displays keywords and frame numbers in this format, when referring to other pages in the database.

Mouse clicks also work on on frames in the tag buffer window, which is useful if you want to go to a page referenced on a tagged frame.

Keypad

The optional keypad to the right of the terminal contains the on-screen numeric keypad, together with a number of icons which transmit standard Viewdata commands or call Viewdata terminal functions. The bottom part of the keypad displays the Online/Offline status of the terminal, the time you have been Online (or the units used), and the current baud rate and data format.



Index

The **Index** option takes you back to the start of the service. This is the index page that you normally see after the welcome page. The index page, is the page from which all other pages are ultimately accessible. This option is useful if you become hopelessly lost within the database. This option transmits the sequence `*0#`.

Retran

The **Retran** option retransmits the current page, free of charge, and leaving any data entered by you intact. It is useful if the page has been corrupted by phone line noise. This option transmits the sequence `*00`.

Back

The **Back** option takes you to the previous page that was displayed. You can normally only go back up to three frames with this command. Any frame charges will be repeated. This option transmits the sequence `*#`.

Leave

The **Leave** option disconnects from the service. If you have some messages waiting for you when you leave Prestel, you will receive a prompt giving you the choice of reading the messages or leaving the service. This option transmits the sequence `*90#`.

Reveal

Some Viewdata pages contain concealed information, and often carry the message `Press Reveal`, to indicate this. The **Reveal** command reveals any concealed information.

Editor

This option opens the Viewdata Editor, described later in this chapter.

Tag

The **Tag** option 'tags' the current frame for easy recall at some later stage. Frame tagging facilities are described later in this chapter.

Show

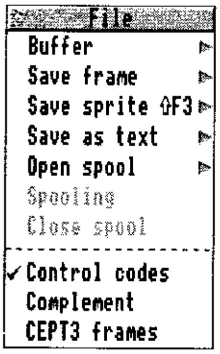
The **Show** option opens a window on the tag buffer, allowing you to manipulate the frames stored in it.

Prestel commands

The list below gives the complete set of Prestel commands at the time of publication. Many of these commands are also available on other Viewdata services.

`*0#`	Return to start of the service.
`*<keyword>#`	Go direct to specified topic.
`*<number>#`	Go direct to specified page number.
`#`	Go forward to next frame or page.
`*L#`	Repeat last keyword, page number or page marker.
`*#`	Go back to previous frame that was displayed.
`.`	Backspace cursor deleting command on bottom line.
`**`	Delete entire command on bottom line.
`*/`	Repeat previous keyword.
`*00`	Retransmit current page.
`?`	Redisplay corrupted command line.
`*09`	Update the current page.
`*02`	Disconnect from a Gateway.
`*90#`	Disconnect from the service.
`*S<name>#`	Set a page mark for the current frame.
`*F<name>#`	Fetch and display the page marked frame.
`*D<name>#`	Delete the specified pagemark.
`*F#`	Show all current page marks.
`*D all#`	Delete all pagemarks.

Saving Viewdata frames



This section describes how to save individual Viewdata frames in various formats, and how to spool all received data to file. If you want to save a number of frames in a single file, please refer to the section on frame tagging later in this chapter. All save and spool operations are chosen from the **File** sub-menu on the main menu. The main menu can be displayed at any time by clicking Menu over the Viewdata terminal window.

Saving frames

Save frame saves the current Viewdata frame. Details of how to load a frame file back into the Viewdata terminal are given in the section on frame tagging, described later in this chapter.

Save sprite saves the current frame as a sprite. The frame can be transferred to disc or to other applications such as Paint.

Save as text saves the current frame as plain ASCII text. This allows the textual contents of the frame to be transferred into a text editor, wordprocessor or DTP package. All graphics characters are ignored, and double-height text is converted to single-height.

If **Complement** is selected, the **Save sprite** option inverts the sprite before saving. This may be useful if you eventually want to print the sprite, because it changes the black terminal background colour to white, which will reduce ribbon wear on dot-matrix printers.

If **CEPT3 frames** is selected, the **Save frame** option saves frames as 1024-byte blocks. Frame files in this format are suitable for loading into some other applications. If **CEPT3 frames** is not selected, frames files are saved as 960-byte blocks, and are compatible with the original Hearsay frame files. If you do not intend to transfer frames into other applications, then it does not matter how this option is set.

Spooling

Open spool spools all received data to the file chosen using the standard save dialogue box. Note that there may be only one spool file open at a time, and that file is common throughout all the terminals in Hearsay.

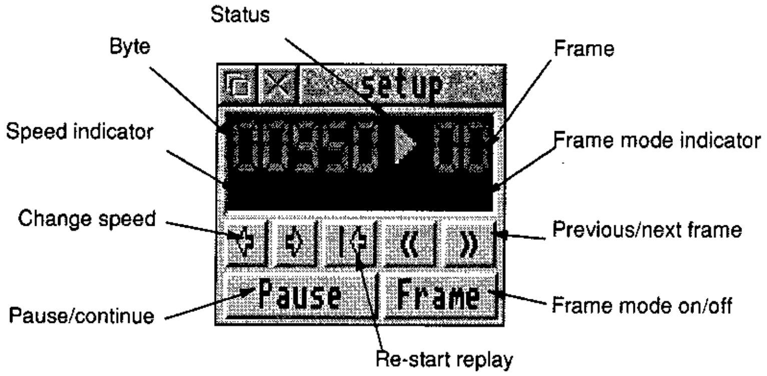
The **Spooling** option switches spooling on and off, and is shaded until a spool file is opened. When you first open a spool file this option will be ticked, indicating that spooling is switched on. Click on **Spooling** to toggle spooling off and on.

Close spool closes the spool file.

If **Control codes** is ticked, both text and Viewdata control codes are spooled to a Viewdata spool file. If **Control codes** is not ticked, only plain text is spooled to an ASCII text file. This option only affects spooling operations.

Replaying spooled files

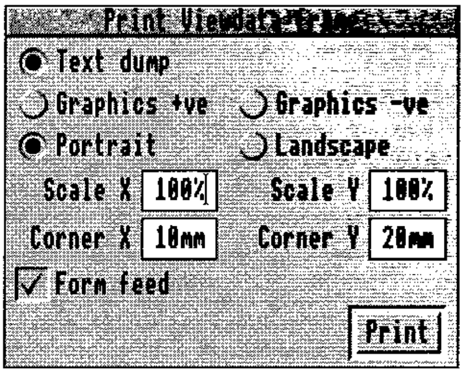
A spooled file may be replayed by pressing Shift and dragging it into the Viewdata terminal window. Replaying a file allows you to re-run a Viewdata session off-line. If the spooled file includes control codes, the replay session will include all the colours and graphics from the original session, otherwise only plain text will be displayed.



Full details of how to use the replay dialogue box are given on page 65. Please note that since the Viewdata terminal is frame-based, it is important to have **Frame mode** switched on. This facility pauses the replay at the end of each frame giving you time to view it. Clicking on **Continue** continues the replay. This facility is toggled on an off using the **Frame** icon, and it's state is indicated in the panel above.

Printing frames

The **Print** option on the main menu opens a dialogue box containing a number of options which control the printing of Viewdata frames.



Print Viewdata frame

● Text dump

○ Graphics +ve ○ Graphics -ve

● Portrait ○ Landscape

Scale X 100% Scale Y 100%

Corner X 10mm Corner Y 20mm

☑ Form feed

Print prints the current Viewdata frame. Whether it does a graphics dump or text dump is determined by the **Text**, **Graphics +ve** and **Graphics -ve** options described below.

If **Text dump** is selected, only a simple text dump is produced by the **Print** option. Please note that text is printed to the printer type specified on the **Printer** sub-menu of the **Choices** menu. This can be via a RISC OS printer driver, or directly to your printer port. Please refer to the chapter *Icon bar menu* for further details.

Please note that if you print using a RISC OS printer driver, the output is not printed immediately, but buffered. The buffer is not printed until a full page is received, or until you force the buffer to be printed, using the **Min memory** option on the icon bar menu. The size of the printed page is defined using the script function `setprinter()` in the `!Custom` auto-run script (please refer to the chapter *CScript* for further details).

If **Graphics +ve** is selected, a graphics screen dump is produced by the **Print** option. This option produces a normal screen dump which causes the normally white text on black terminal background to be printed white on black. This may not be suitable for some printers, due to the large amounts of black background that might be printed. You must have a suitable RISC OS printer driver loaded before you can do any graphics printing.

If **Graphics -ve** is selected, a graphics screen dump is produced by the **Print** option. It produces a negative screen dump which causes the normally white text on black terminal background, to be printed as black text on white paper.

If **Portrait** is selected, the frame is printed in the same orientation that it appears on the screen. If **Landscape** is selected, the frame is turned through 90 degrees so that it is printed on its side.

Scale X and **Scale Y** allow you to adjust the scaling of the printed frame. The default values are 100%, which print the frame at actual size.

Corner X and **Corner Y** specify where the bottom left-hand corner of the frame will be positioned on the paper. The default values are 10mm and 20mm which will position the frame in the bottom left-hand corner of the printable area of the paper.

If **Form feed** is selected, a form feed will be printed after each frame.

Frame tagging

Whilst it is easy to to follow a series of indexes through a Viewdata database to a particular frame, it is quite difficult to find you way back to other frames you noticed on the way. The **Back** option helps to some extent, but since you can only step back through the previous 3 frames, it is somewhat limited.

Frame tagging is a simple solution to these problems. It enables you to put 'tags' on any frames that you may wish to return to at a later stage. The number of frames that you can tag, is limited only by the memory available on your system (each tagged frame uses approximately 1K). Although frame tagging is a feature common in many Viewdata terminals, this implementation has a number of special features:

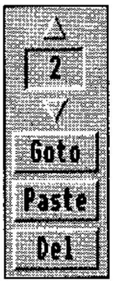
- When you tag a frame, the complete image is placed in a buffer. You can open a window on that buffer and step through the tagged frames viewing them one at a time. Hearsay allows a number of named buffers to be created.
- You can select a frame in the buffer, and 'goto' it. This means that Hearsay will automatically type in the frame number for you, and take you directly to that frame in the database.
- The entire contents of the tag buffer may be saved to disc, and subsequently re-loaded at a later date. Once re-loaded, you can 'goto' any of the frames in the buffer. This means that frames tagged in one session, may be returned to with ease in another.

Tagging frames

To tag the current frame in the Viewdata terminal, click on **Tag** on the Viewdata keypad. If you have not specifically opened a named buffer for the tagged frames, Hearsay will open a default buffer called `Buffer1`. The buffer name is displayed in the box to the right of the tag button.

Clicking on **Tag** will copy the current frame into `Buffer1`, and any subsequent clicks on **Tag** will append frames to the end of the buffer. The number shown after the buffer name, gives the position of the tagged frame in the buffer.

To view the contents of `Buffer1`, click on **Show** on the keypad. A new window will be opened, displaying the last frame tagged. A small tools panel is displayed to the left of the tag window.



Click on the up and down arrows to step up and down through the tagged frames. The number between the arrows, gives the index of the currently displayed frame in the buffer. There are three further options on this toolbox:

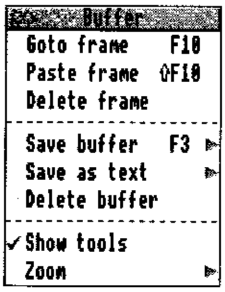
Goto takes you directly to the currently displayed frame. It does this by sending the appropriate command and page number to the database. There will be a slight delay while the chosen frame is transmitted. If you click Select on **Goto**, the tag window will remain displayed after the operation, otherwise if you click Adjust, the tag window will be closed.

Paste simply transfers the currently displayed frame to the viewdata terminal, overwriting anything that is already there. It has two uses:

1. Frame tagging may be used offline as a general purpose store for frames being edited by the Viewdata editor. As frames are edited they may be tagged, and then recalled when required using the **Paste** option. The **Goto** option cannot be used here, since you cannot 'goto' a frame if you are offline, or if the frame has no frame number.
2. It may be used online to overlay mailbox frames prepared offline, onto the Viewdata terminal for transmission using **Frame send**. Further details of this are given later in this chapter.

Del deletes the current displayed frame from the buffer, shifting any following frames down to fill the gap. You will be asked for confirmation before deletion

Tag window menu



Click Menu over the tag window to display the tag window menu.

The first three options on this menu **Goto, Paste, and Del** perform the same function as those options on the tools panel described above.

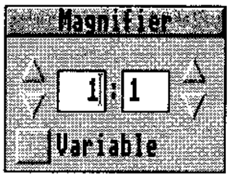
Save buffer saves all the frames in the buffer. The default frame file name offered is the same as the buffer name.

Save as text save the buffer as plain ASCII text. This allows the textual contents of all the frames in the buffer to be transferred into a text editor, wordprocessor or DTP package. All graphics characters are ignored, and double-height text is converted to single-height.

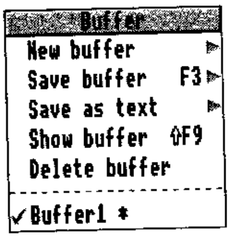
Delete buffer deletes the buffer and all the frames stored in it. Since this operation cannot be undone, a warning message is displayed.

Show tools allows you switch the tools panel on and off.

Zoom opens the standard magnifier dialogue box, and is used for scaling the Viewdata terminal.



The magnification is expressed as a ratio; for example 2:1 means twice normal size, 3:4 means three-quarters normal size, and so on. You can change the magnification ratio by clicking on the arrows, or by typing in a new value. To remove the magnifier dialogue box, press Escape or move to a different menu item. If **Variable** is selected, the terminal is scaled to the size of the window.



The Buffer menu

The **Buffer** menu is found on the **File** sub-menu on the main menu. It allows you to create new tag buffers with names of your choice, and manipulate them.

New buffer creates a new tag buffer with the name specified. A warning will be given if a buffer already exists with that name. The new buffer name is added to the list at the bottom of the menu, and is ticked to indicate that it is the currently selected buffer. The tick is removed from the previously selected buffer.

An asterisk after the buffer name indicates that the buffer has been modified but not saved.

Save buffer saves all the frames in the currently selected buffer.

Save as text saves the textual contents of all the frames in the currently selected buffer, as a plain ASCII text file. All graphics characters are ignored, and double-height text is converted to single-height.

Show buffer opens a window on the currently selected buffer.

Delete buffer deletes the currently selected buffer and all the frames stored in it. Since this operation cannot be undone, a warning message is displayed.

In Hearsay, all operations on tag buffers apply to the currently selected buffer. The selected buffer is the one that is ticked at the bottom of the menu. To select another buffer simply click on its name in the list at the bottom of the menu. Viewdata buffers remain in existence until the buffer is deleted, Hearsay is quitted, or the **Min memory** option is selected (only if the terminal window is closed). In all three cases you will be warned before the buffer is removed and the frames lost.

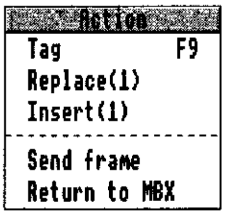
Re-loading frame files

Hearsay frame files are identified by the following icon:

CEPT 3 frame files are identified by the following icon:

Either of these file types may be loaded by simply double-clicking on it. The frames in the file will be loaded into a buffer with the same name as the frame file, and a window on that buffer opened (loading Hearsay, and opening the Viewdata terminal if necessary). The same sequence occurs if a frame file is dragged onto the Hearsay icon on the icon bar. However, if a frame file is dragged into the tag window, the frames are appended to the end of the existing frames in that tag buffer.

Inserting & replacing frames



The **Action** sub-menu on the main menu provides options for inserting frames into the tag buffer, and replacing existing frames in the buffer.

Tag appends the current frame to the currently selected tag buffer. This option is equivalent to the **Tag** option on the Viewdata keypad.

Replace replaces the current frame in the tag buffer with the frame in the Viewdata terminal. The index number of the current frame is given in brackets.

Insert inserts the frame in the Viewdata terminal into the tag buffer before the current frame. The index number of the current frame is given in brackets.

The other options on this menu **Send frame** and **Return to MBX**, are described later in this chapter.

Sending messages

One of the popular features of Prestel (and most other Viewdata services) is the mailbox facility, which allows you to send messages to other Prestel users. To send a message you must know the mailbox number of the recipient, and the message must be typed into a mailbox reply frame. The following text describes how to send a message whilst on-line, but it is possible to prepare a message offline, and only call Prestel when you are ready to send it. The second method can save you unnecessary time online, and is described later in this chapter under the section entitled *Frame Send.*

The first thing to do is type `*MBX#` which will take you to the mailbox section of Prestel. Now select option 1 to display the Address Frame containing two empty fields which you should fill in. The flashing cursor will be correctly positioned at the start of the first field labelled `To:`, and you should type in the recipients mailbox number. When you have finished, press Return (#). The cursor will automatically move to the next field labelled `Subject:`, which may be used to give brief details of the subject of the message you are to send. If you do not want to fill in a field, just press Return to move to next one. After filling in these two fields press Return again for a blank screen into which you may type your message. When you have finished press Return, and follow the on-screen instructions giving details of how to send your message. At this stage you will be given the opportunity to cancel the operation.

When filling in fields you may use the cursor keys and the Delete key in the normal way, to correct mistakes. In addition you may enter Viewdata attributes to give your messages colour. The colour codes are entered by pressing Esc followed by a character:

A	red alphanumeric
B	green alphanumeric
C	yellow alphanumeric
D	blue alphanumeric
E	magenta alphanumeric
F	cyan alphanumeric
G	white alphanumeric

Other attributes can also be entered by pressing Esc followed by a special character, and a full list is given on Prestel. However, if you are intending to send messages containing many Viewdata attributes, it is recommended that you use the Viewdata editor to prepare the message offline, and then use the **Frame send** function to transmit it. Full details of the editor and frame send functions, are given later in this chapter.

In addition to the above attributes, Shift-£ generates the solid block character, and Shift-Copy transmits the character under the cursor.

The following table gives a list of editing commands that can be used whilst editing messages. The commands are introduced by pressing `@` followed by a lower-case letter.

`@i`	insert a space into the line
`@d`	delete character and close up gap
`@v`	verify - redisplay frame as received at host
`@r`	Carriage Return to next line
`@f`	move forward to next word space
`@b`	move back to previous word space
`@e`	move to end of text
`@s`	print * in field
`@h`	print # in field
`@@`	print `@` in field
`@o`	turn off word-wrap
`@w`	turn on word-wrap
`@)`	disable use of `@` as Esc
`Esc(`	enable `@` as Esc

The mouse buttons may be used as follows:

single-click Select	transmit character under pointer
single-click Adjust	send Return character
double-click Select	home cursor to pointer position

Reply to MBX

Reply to MBX is a time saving feature which allows a mailbox number to be read directly from the Viewdata terminal, and automatically entered into a mailbox reply frame. If you wish to send a message to a user whose mailbox number is displayed in the terminal, simply double-click Adjust on that number. The number will be read from the screen and assigned to the macro {mbx}. Next the macro defined by **MBX reply route** on the **Misc** dialogue box, is expanded. By default, this macro is defined as follows:

```
*MBX_1{mbx}__
```

Where _ is translated to #, and:

`*MBX_`	call mailbox menu frame
`1`	select 1 to send mailbox
`{mbx}`	enter the mbx number read from the screen
`_`	send # to ignore date field
`_`	send # to ignore subject field

After using the **Reply to MBX** facility you may use **Return to MBX** on the **Action** menu, which will return you to your original frame.

Send frame

This option is used to send mailbox messages that have been prepared offline to the host computer. It is generally easier and quicker than typing in the message whilst online (described earlier in this chapter). The function operates by sending each character of a prepared frame, which has been loaded onto the screen over the current Prestel frame. The following is a step by step example:

1. Go online to Prestel and tag the blank mailbox reply frame i.e. the frame into which you would normally type your message. To do this type `*MBX#`, then press `1` followed by Return twice.
2. Leave Prestel, and go offline.
3. Transfer the tagged frame in the tag buffer to the screen using **Paste**.
4. Now enter the Viewdata Editor and fill in the message field. Remember, that to use the editor offline you must select **Local** on the **Line mode** menu. It is important that your message starts at the correct position in the frame, so you may need to send a message online first, and remember where the message starts. The frame may contain coloured text and graphics. Finally, always remember to end the message with a `#`.
5. Now tag the prepared frame. At this stage you make wish to save the tagged buffer to disc, for future reference.
6. To send the message switch off **Local** mode, and call Prestel. When you are on-line, type `*MBX#`, followed by `1` to go to the address frame. Enter the recipients mailbox number and a subject then press Return. At this stage, the frame displayed should be a blank message frame identical to the one you filled in offline.
7. Choose **Show** on the viewdata keypad, and transfer your prepared frame from the tag buffer to the screen using **Paste**.
8. Finally select **Send frame** from the **Action** menu. As each character is transmitted to Prestel, you will see the cursor move through the fields, until the end of the frame is reached. Now follow the on-screen instructions to send the message or cancel the operation.

If your message is corrupted during transmission, click on **Cancel** in the **Send frame** dialogue box, and move the cursor (using the cursor keys) to the correct start of the message frame. Now transfer your prepared frame from the tag buffer, and use Send frame again. See also the setup options on **Misc** dialogue box.

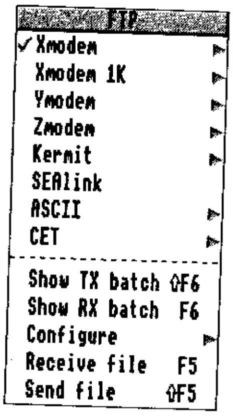
Telesoftware downloading

Prestel and most other Viewdata systems provide a software downloading service for different computers, including the Acorn RISC machines. *Telesoftware* downloading is the process of transferring programs or files from Prestel to your computer. The downloader receives files in CET telesoftware format, and writes them to disc for subsequent execution.

The first thing to do, is locate the program you want to download. You can download almost any program, but generally, unless it is specifically for the Acorn RISC machines it will not run. Next, locate the frame which contains the program header. There are usually a few frames of descriptive text before the header frame, but it can normally be identified by a message requesting that you start the download procedure, usually the frame number has the suffix ‘c’.

You may notice whilst locating the program header, instructions telling you how to run the program after downloading is finished. Make a note of these, since some programs are encoded for downloading, and do not always run in the expected manner. In particular, many programs are archived, and must be de-archived before they can be run (please refer to *Appendix D* for details of SparkPlug, a de-archiving application).

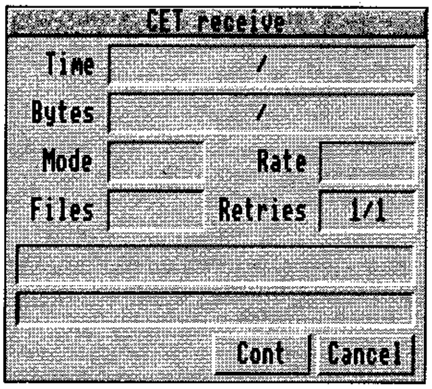
Notice also that some programs are free, and some have charges. If you download charged frames, the cost will be automatically added to your Prestel bill.



When you are ready to start the download, choose **CET** on the **File Transfer** sub-menu of the main menu. **CET** is the protocol used for telesoftware downloading. Now click **Receive file** on the same menu to start the download.

Telesoftware is downloaded as a number of Viewdata frames, which appear in the terminal window as they are received. The downloaded file is automatically given a name by host computer, and is saved in a special directory called `RXBatch`, which is located in the `!Hearsay` application directory.

During downloading the standard file transfer status box is displayed, giving information about the progress of the download. In the event of a frame being corrupted by line noise, the downloader will automatically request re-transmission of the frame, and it will continue to do so until the frame is received uncorrupted, or until you select **Cancel** to abandon the download. The number of retries is displayed in the status box.



Time /
Bytes /
Mode
Rate
Files
Retries 1/1

You may cancel the download at any time by clicking on **Cancel**. After cancelling, you will be left with a telesoftware frame displayed on the screen; simply select **Index** from the keypad or enter a frame number to continue.

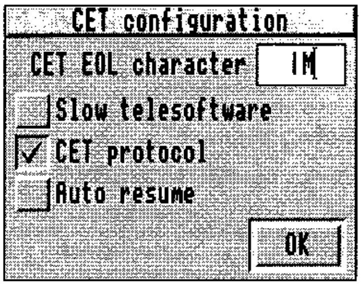
When downloading is complete, the received file may be viewed by choosing **Show RX Batch** on the **File transfer** menu. This opens a window in the style of a standard directory display, showing all the files that have been downloaded during the current session.

You can run downloaded telesoftware by double-clicking on its icon in the RX batch display, but we recommend that you move downloaded files to another directory to prevent the batch directory becoming full. To do this, drag the files from the batch to a suitable directory, then remove the file from the batch using the **Remove** option. You can do this in one operation by holding down Shift whilst dragging i.e. moving the file. Please refer to the chapter *File transfer* for further details about batch file transfer.

Before attempting to run downloaded software, remember to de-archive the file if necessary, or follow any other special running instructions given. Please refer to *Appendix D* for details of de-archiving files.

Telesoftware configuration

The CET dialogue box on the **File transfer** menu allows the CET downloader to be specially configured for non-standard Viewdata services. You will not normally need to change any of these options if you are using Prestel.



CET configuration

CET EOL character |M

□ Slow telesoftware

☑ CET protocol

□ Auto resume

CET EOL specifies the codes that the downloader should use to mark the end of lines. Normally it should be set to `|M`, but if you are downloading software for running on another make of computer, you may need to change it. Try `|M|J` for a PC, or `|J` for a Unix machine.

Slow telesoftware causes the downloader to operate at a slower speed.

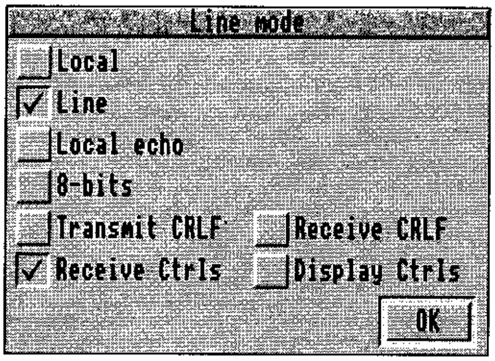
CET protocol is normally selected and causes the downloader to strictly adhere to the CET protocol. For some services this option may need to be de-selected.

The **Auto resume** option allows you to resume a CET download at any point in the file. Typically it is used to continue a download after it has terminated due to line problems or cancellation. **Auto resume** occurs when you download a file whose name matches a file in the RX batch. Instead of overwriting the existing file, the download continues from the end of the existing file.

Terminal setup menu

Line mode

All options on this dialogue box are described on page 68.



Line mode

□ Local

☑ Line

□ Local echo

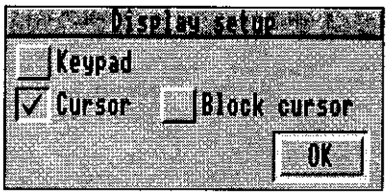
□ 8-bits

□ Transmit CRLF □ Receive CRLF

☑ Receive Ctrls □ Display Ctrls

Display

The **Display** dialogue box provides options that allow you to control the Viewdata terminal display.



Display setup

□ Keypad

☑ Cursor □ Block cursor

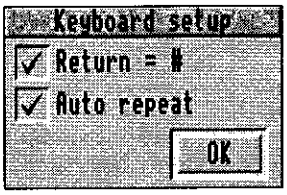
Keypad switches on and off the Viewdata keypad.

Cursor switches the cursor on and off.

If **Block cursor** is selected, the flashing underscore cursor changes to a flashing block. You may find this preferable since the block cursor is easier to see.

Keyboard

The **Keyboard** dialogue box provides a number of options concerning keyboard operation.



Keyboard setup

☑ Return = #

☑ Auto repeat

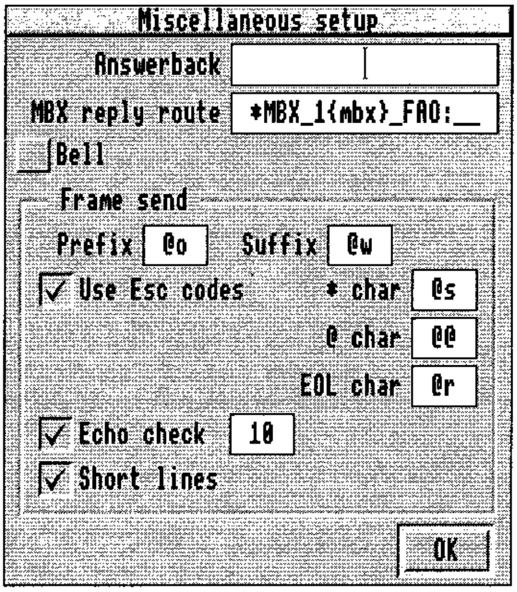
Return = # causes the Return key to send a # character. This is very useful because # is commonly used in a Viewdata terminal.

Auto repeat determines if keys auto repeat.

Macros opens a dialogue box which allows you to re-define virtually all the keys on the keyboard, and to define macros that may be used in those definitions. A full description of this facility is given in the chapter *Macros.*

Misc

This dialogue box contains a number of miscellaneous options concerned with the Viewdata terminal.



Miscellaneous setup

Answerback

MBX reply route *MBX_1{mbx}_FAO:__

□ Bell

Frame send

Prefix @o Suffix @w

☑ Use Esc codes * char @s

@ char @@

EOL char @r

☑ Echo check 10

☑ Short lines

Answerback allows you to define a string which identifies your terminal. The host system may request this string.

MBX reply route allows you to define a macro which is automatically expanded when the **Reply to MBX** function is used. Typically it is used to call up a blank mailbox reply frame and insert the recipients mailbox number. Please refer to page 92 for further details.

If **Bell** is selected, Hearsay will sound a beep when it receives ASCII code 7, otherwise this code is ignored..

The following options configure the frame send function described earlier. In normal use you should not need to alter any of these values, however they may prove useful on non-standard Viewdata systems.

Frame send prefix specifies the string that is transmitted just before the frame send proceeds. Normally it is set to `@o`, which switches off the Prestel mailbox word-wrap facility during the frame send.

Frame send suffix specifies the string transmitted after the frame send has completed. Normally it is set to `@w` which turns word-wrap on.

Use Escape seq. defines escape sequences for characters which have a special purpose in the Viewdata terminal. During the frame send, the escape sequences are transmitted in place of the normal characters.

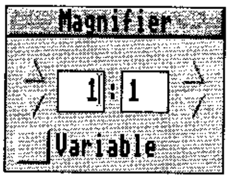
*** char** is set to the escape sequence for * (normally @s)
@ char is set to the escape sequence for @ (normally @@)
EOL char is set to the escape sequence for CR (normally @r)

If **Short lines** is selected, trailing spaces on each line of the frame are not transmitted, thus speeding up the operation. Instead of transmitting trailing spaces, the `EOL char` is transmitted to force a new line. If **Short lines** is not selected, the entire contents of each frame line are transmitted, including trailing spaces. The key press Ctrl-Return sends the `EOL char`.

If **Echo check** is selected, Hearsay checks the echoed character received against the original character transmitted. If they are not the same, the original character is re-transmitted. If **Echo check** is not selected, the frame send operates more quickly, but on a poor line corrupted characters may be transmitted. **Echo check** also allows you to specify a delay between characters transmitted. The lower the delay specified, the quicker the frame send will be, albeit with a greater risk of corruption. The delay should be in the range 0 to 99 centi-seconds.

Zoom

Zoom opens the standard magnifier dialogue box, and is used for scaling the size of the Viewdata terminal.



The magnification is expressed as a ratio; for example 2:1 means twice normal size, 3:4 means three-quarters normal size, and so on. You can change the magnification ration by clicking on the arrows, or by typing in a new value. To remove the magnifier dialogue box, press Escape or move to a different menu item. If **Variable** is selected, the terminal is scaled to the size of the window.

Macros

Macros opens a dialogue box which allows you to re-define virtually all the keys on the keyboard, and to define macros that may be used in those definitions. Full information is given in the chapter *Macros*.

Reset

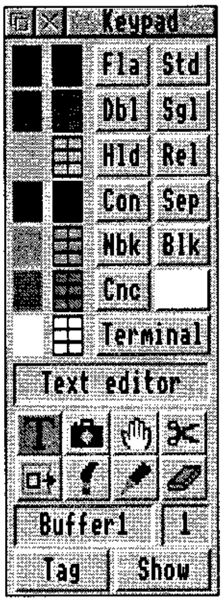
Reset resets the terminal, clears the screen and homes the cursor.

Viewdata editor

The Viewdata Editor in Hearsay allows the user to generate Viewdata frames primarily for sending messages using the frame send facility. It can, however, be used to create Viewdata frames for any purpose including things unrelated to communications. This editor is primarily an offline editor, and the description that follows assumes you are using it offline, but it can be used online with certain restrictions (see later). If you are using it offline, you must set the terminal to **Local** mode on the the **Line mode** menu. It is beyond the scope of this user guide to explain fully the workings of teletext graphics, but a useful explanation is given in your computer user guide.

To call the editor click on the **Editor** icon on the Viewdata terminal keypad. Any text or graphics in the terminal window will be preserved, but the keypad will change to display a number of new icons which are used to select editor facilities.

To leave the editor click on the **Term** icon. The Viewdata terminal is re-entered, preserving the edited frame.



Simple cursor movement

Characters typed at the keyboard appear at the current cursor position. The cursor is defined as a flashing rectangle outline. To move the cursor to a new position, simply move the mouse pointer to the required position and press one of the mouse buttons. Alternatively you can use the cursor control keys for cursor movement. The Delete key deletes the character to the left of the cursor, and Return moves the cursor to the start of the next line.

The white box above the **Term** icon on the keypad shows the Viewdata character or attribute at the mouse pointer position. Viewdata attributes are identified by the same mnemonics as those listed overleaf.

Viewdata text colours

At the top right-hand side of the keypad are two columns of coloured icons. The first column represents the Viewdata text colours that are available. Simply click on the colour you require and the relevant colour code will be placed at the cursor position. The effect of these codes is to change the colour of all subsequent text up to the end of the line, or until another colour code is reached. Remember that each colour code occupies a character position, which will appear as a space on the screen.

Viewdata graphics colours

The second column of coloured icons represents the graphics colours available. Clicking on one of these icons, places the relevant graphics colour code at the cursor position. The effect of graphics colour codes is to cause all subsequent lower-case characters to be displayed as graphics characters in the relevant colour. This remains in effect up to the end of the line, or until another text or graphics code is reached. A table showing the graphics shape are produced by each lower-case character, is given in your computer user guide.

If you want to design complex screens using teletext graphics, then you will find the Pixel Editor (described later in this chapter) indispensable.

Other Viewdata effects

Also at the top right-hand side of the keypad, are a number of icons containing mnemonic codes. These are:

Fla	flash	**Std**	steady
Dbl	double height	**Sgl**	single height
Hld	hold graphics	**Rel**	release graphics
Con	contiguous graphics	**Sep**	separated graphics
Nbk	new background	**Blk**	black background
Cnc	conceal		

Please refer to your computer user guide for full details of these effects.

Editor functions

The following functions are available in the Viewdata editor by clicking on the appropriate icon on the keypad.

Text edit

This function is used to edit text.

Copy block

This function allows you to copy any marked area of the editor screen. To use the function, click on **Copy** and position the mouse pointer at the top left hand corner of the rectangle you wish to copy. Hold down the left mouse button, and drag the mouse until the bounding rectangle coincides with the area you wish to copy. Now release the mouse button and move the block to the required position. Finally press the left-hand mouse button to paste the block back on the screen. Press Escape or click Adjust to cancel the operation.

Please note that you can only mark blocks on character boundaries. If the pointer is not on character boundaries, the boundaries above and to the left of the pointer position are used.

This function has only local effect if used online.

Move block

This function moves a marked area of the screen to a new position. To use the function, click on **Move** and position the mouse pointer at the top left hand corner of the rectangle you wish to move. Hold down the left mouse button, and drag the mouse until the 'bounding' rectangle coincides with the area you wish to move. Now release the mouse button and move the block to the required position. Finally press the left-hand mouse button to paste the block back on the screen. Press Escape or click Adjust to cancel the operation after you have marked a block.

This function has only local effect if used online.

Delete block

This function simply deletes a marked block. Mark the block in the usual way, then click Select to delete it. If you change your mind after selecting this function, press Escape or click Adjust to cancel. This function has only local effect if used online.

Save block

This function stores a marked area for re-call using the **Paste** function. It is used for transferring marked sections from one frame to another. This function has only local effect if used online.

Paste

This function re-calls a block previously stored using the **Save** function. The re-called block may be pasted any number of times onto the current screen, until the function is cancelled with the Adjust button. This function has only local effect if used online.

Pixel editor

This function allows you to edit individual graphics *pixels*. It displays a grid on the screen over those parts of the screen that can contain Viewdata graphics i.e. those which commence with a Viewdata graphics code. If you select this option and no grid appears, it means that there are no graphics codes on the screen. In this case you should put graphics codes at the left-hand edge of the areas into which you wish to put graphics.

When the pixel grid is displayed, the mouse pointer can be moved over the grid, and a pixel switched on by clicking Select. Pixels are switched off by clicking Adjust. While the pixel grid is displayed, normal cursor movement and text editing is disabled, but most other editor functions are available. This function has only local effect if used online.

Clear screen

This function clears the screen. A warning prompt is given.

Special keyboard functions

Ctrl ← Delete character under cursor, shifting all characters to the right of the cursor left one position (same as Copy). Has only local effect.

Ctrl → Insert a space at cursor position, shifting all characters after the cursor right one position. Has only local effect.

Ctrl ↑ Delete current line, shifting all lines below it up one line. Has only local effect.

Ctrl ↓ Insert blank line at cursor position, shifting all lines below it down one line. Up to 10 lines are buffered if they are shifted off the bottom of the screen, and may be re-called using Ctrl ↑ described above.

10 Tektronix terminal

This terminal is an emulation of Tektronix 4010 and 4105 graphics terminals with some 4107 extensions.

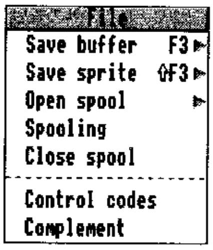
If you require a Tektronix 4010 emulation you should select the **Tek 4010** and de-select the **Text plane** options on the **Misc** dialogue box.

If you require a Tektronix 4105 emulation, de-select **Tek 4010** and select **Text plane**.

The Tektronix menu options are described below.

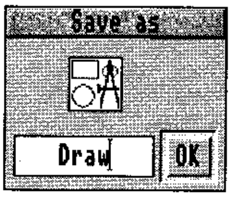
Please note that the **Select** menu has no purpose in this emulation and is always shaded.

File menu



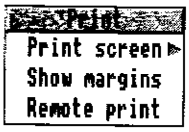
The **File** menu provides a number of options for saving Tektronix graphics and spooling received data.

Save buffer saves the current Tektronix graphics screen in Draw file format. It may be saved to disc or directly into Draw or other applications that can import Draw format files.



The **Spool**, **Control codes** and **Complement** options are identical to those described for the text terminals, so please refer to page 64 for further details.

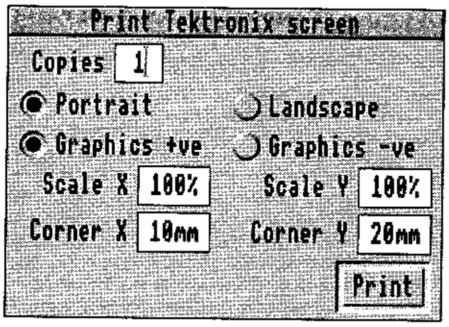
Print menu



The **Print** menu controls printing from the Tektronix terminal. The **Print** menu from the text plane of the terminal provides the same options as the text terminals (see page 67). The **Print** menu from the graphics window of the terminal provides options for dumping the contents of the graphics window. These options are described below.

Print screen

The **Print screen** option opens a dialogue box which controls the printing of the graphics window. You must have a suitable RISC OS printer driver loaded before you can use this option.



Print Tektronix screen

Copies 1

● Portrait ○ Landscape

● Graphics +ve ○ Graphics -ve

Scale X 100% Scale Y 100%

Corner X 10mm Corner Y 20mm

Copies specifies the number of copies to be printed.

If **Portrait** is selected, the frame is printed in the same orientation that it appears on the screen. If **Landscape** is selected, the frame is printed turned through 90 degrees so that it is printed on its side.

If **Graphics +ve** is selected, a normal screen dump is produced i.e. white text on black background. This may not be suitable for some printers due to the large amounts of black that might be printed.

If **Graphics -ve** is selected, a negative screen dump is produced i.e.black text on white background. This will be more suitable for many printers, especially dot-matrix printers.

Scale X and **Scale Y** allow you to adjust the scaling of the printed frame. The default values are 100%, which print the frame at actual size.

Corner X and **Corner Y** specify where the bottom left-hand corner of the frame will be positioned on the paper. The default values are 10mm and 20mm which will position the frame in the bottom left-hand corner of the printable area of the paper.

Click on **Print** to print the graphics screen.

Show margins

Most printers cannot print right up to the edges of the paper. If **Show margins** is selected, grey borders are displayed around the edges of the terminal representing the area that cannot be printed. Since these margins are derived from the current printer driver, the print margin will only be displayed if a printer driver is loaded. **Show margins** is intelligent and takes into account the orientation, scaling and print position specified in the **Print screen** dialogue box.

Remote print

If **Remote print** is selected, Hearsay will act upon escape sequences received from the host which control the printer. **Remote print** should be selected if you want to allow the host to do automatic screen dumps.

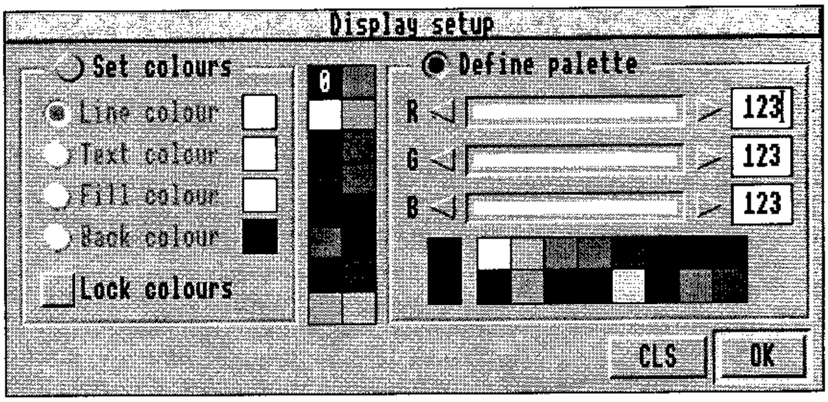
Terminal setup menu

Line mode

All options on this dialogue box are described on page 68.

Display

The **Display** dialogue box allows you to control the Tektronix display.



Display setup

○ Set colours
● Line colour
○ Text colour
○ Fill colour
○ Back colour
□ Lock colours

● Define palette
R 123
G 123
B 123

Click on Set colours to set **Line colour**, **Text colour**, **Fill colour** and **Background colour.** Choose the type of object you wish to set, then choose the colour you require from the palette in the centre of the dialogue box.

Lock colours locks the palette so that it cannot be changed by the host.

Set colours allows you to re-define the palette of 16 colours shown in the centre of the dialogue box. Choose from the palette the colour you wish to define, then use the standard sliders etc. on the right of the box to define the colour as required.

Please note that not every possible colour shade can be shown on the screen, so the best approximation for the current screen mode is used. The exact colour is recorded internally so that it can be displayed as accurately as possible in other screen modes or when printed.

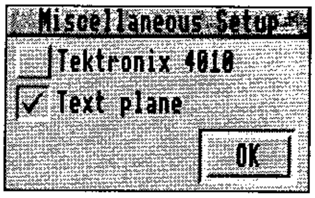
CLS clears the terminal window.

Keyboard

The Keyboard dialogue box in the Tektronix terminal is identical to that described for the text terminals on page 71.

Misc

This dialogue box contains two miscellaneous options concerned with the Tektronix terminal.



□ Tektronix 4010

☑ Text plane

Tek 4010 should be selected if you want Hearsay to emulate a Tektronix 4010 graphics terminal. If this is not selected, Hearsay emulates a Tektronix 4105 terminal.

If **Text plane** is selected, text is displayed in a separate VT102 scrolling window, and graphics in the Tektronix graphics window. This is the correct operation for Tektronix 4105 emulation. This option should not be selected if you wish to emulate a Tektronix 4010 terminal.

Zoom

Zoom opens the standard magnifier dialogue box, and is used for scaling the Tektronix graphics window (see page 75 for details).

Macros

Macros opens a dialogue box which allows you to re-define virtually all the keys on the keyboard, and to define macros that may be used in those definitions. Full information is given in the chapter *Macros*.

Reset

The **Reset** option resets the Tektronix graphics terminal.

11 Macros

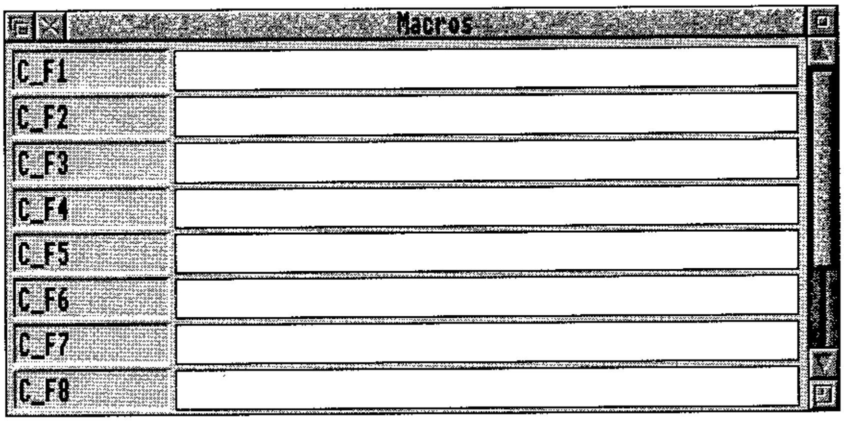
The Hearsay macro facility allows virtually all the keys on the keyboard to be redefined, or have complex expressions attached to them.

Two type of macros may be defined; Key macros and User macros.

- Key macros represent keys on the keyboard allowing them to be defined. Virtually all the keys may be redefined including special key combinations using Ctrl, Shift etc.
- User macros allow common constants to be given simple textual names.

Both types of macros may be used in the definitions of other macros.

An additional facility allows a macro to be defined which will be accessible directly from a menu option from the Script menu. This menu may be chosen to appear instead of the main Hearsay menu, allowing the software to be completely customised for special applications (see page 60 for further details).



Choose **Macros** from the **Terminal setup** menu of any terminal to display the macros dialogue box. The left-hand column of this box shows the names of the macros, and the right-hand side, the definitions.

Key macros

For your convenience the key macros Ctrl-F1 to Ctrl-F11 have been created, and are shown on the left. Their definitions, which are initially blank, are shown on the right. The C_ prefix to the key names signifies that the left hand Ctrl key key is to be held down at the same time as function key. You can define other key macros by clicking Menu over the dialogue box, and choosing **New**. Type in the name of the key you wish to define.

The names of the keys that may be defined are listed below:

f1
f2
f3
f4
f5
f6
f7
f8
f9
f10
f11
f12

`
!
@
#
$
%
^
&
*
(
)
-

=
£
BS
a
b
c
d
e
f
g
h
i

j
k
l
m
n
o
p
q
r
s
t
u

v
w
x
y
z
Space
[
]
\
:
"
,

.
/
Escape
Print
SLoc
Break
Insert
Home
PageUp
Delete
Copy
PageDwn

Tab
Return
CUp
CLeft
CDown
CRight
NLock
K/
K*
K#
K-
K+

Enter
K0
K.
K1
K2
K2
K4
K5
K6
K7
K8
K9

Please note that the keys preceded by K refer to keys on the numeric keypad.

These keys may be preceded by any of the following sequences:

S_	key plus Shift
C_	key plus Ctrl (the left-hand Ctrl key only)
A_	key plus Action (the right-hand Ctrl key)
CS_	key plus Ctrl Shift
AS_	key plus Action Shift
M_	show macro on the User menu

If a macro name is preceded by M_ it is listed on the User sub-menu on the CScript menu. The name appears on the menu without the M_ prefix, and allows you to build a menu of user functions.

Please refer to page 60 for further details of the **User** menu and how it can be displayed.

User macros

The macros described earlier, represent keys on the keyboard and are called key macros. However, any macro defined that does not match a key name in the table above, is called a user macro. This type of macro can be used in other macro definitions, and is useful for defining constants that you may wish to use several times.

Definitions

To define any macro, click Select in the corresponding box on the right-hand side of the macro name, and type in a string. The string may contain all the standard ASCII characters, but must not begin with { or <. If you wish to start the string with either of these characters, you must use the alternative sequences |{ and |<. In addition to normal printable characters, you can introduce any other ASCII character using the standard escape sequences given below.

<table>
<tr><td>0</td><td>|@</td><td>8</td><td>|H</td><td>16</td><td>|P2</td><td>4</td><td>|X</td></tr>
<tr><td>1</td><td>|A</td><td>9</td><td>|I</td><td>17</td><td>|Q</td><td>25</td><td>|Y</td></tr>
<tr><td>2</td><td>|B</td><td>10</td><td>|J</td><td>18</td><td>|R</td><td>26</td><td>|Z</td></tr>
<tr><td>3</td><td>|C</td><td>11</td><td>|K</td><td>19</td><td>|S</td><td>27</td><td>|[</td></tr>
<tr><td>4</td><td>|D</td><td>12</td><td>|L</td><td>20</td><td>|T</td><td>28</td><td>|\</td></tr>
<tr><td>5</td><td>|E</td><td>13</td><td>|M</td><td>21</td><td>|U</td><td>29</td><td>|]</td></tr>
<tr><td>6</td><td>|F</td><td>14</td><td>|N</td><td>22</td><td>|V</td><td>30</td><td>|^</td></tr>
<tr><td>7</td><td>|G</td><td>15</td><td>|O</td><td>23</td><td>|W</td><td>31</td><td>|_</td></tr>
</table>

ASCII code 127 is represented by |?, and characters 128 to 255 are represented by |! followed by the low-bit code.

As usual the sequence || may be used to represent |.

Examples

<table>
<tr><th>Key/Macro</th><th>Definition</th><th>Effect</th></tr>
<tr><td>C_F1</td><td>password</td><td>defines key Ctrl-F1 == password</td></tr>
<tr><td>C_F5</td><td>logon|M</td><td>defines key Ctrl F5 == logon + CR</td></tr>
<tr><td>S_k0</td><td>Menu|M</td><td>defines Shift-0 (keypad) == Menu + CR</td></tr>
<tr><td>#</td><td>£</td><td>defines # == £</td></tr>
<tr><td>log</td><td>AB38600X</td><td>defines user macro log == AB38600X</td></tr>
<tr><td>S_F2</td><td>log|M</td><td>defines Shift-F2 == AB38600X + CR</td></tr>
</table>

Advanced definitions

In the examples given above, it has been shown that macros definitions may contain all the standard printable ASCII characters, plus any control codes using the | convention.

However, definitions may also contain operating system expansions enclosed in angle brackets < >, and Hearsay expansions enclosed in braces { }.

`<string>` A string within angle brackets is evaluated using OS_ReadVarVal. This allows you to place things like the following in a definition.

```
<Sys$Time>
```

`<number>` A number within angle brackets is evaluated using OS_ReadUnsigned. This allows the following numbers to be evaluated:

```
<1>
<&FF>
<2_10111010>
```

`{token}` A token such as NUL, SOH etc. Tokens are case independent. A full list of tokens is given in *Appendix A*.

`{key macro}` A key macro corresponding to a key press such as `S_F1`, `CS_F5`, `Insert` etc. These keys will be re-evaluated, unless preceded by a single quote, in which case the original operation of the key is invoked.

For example, to invoke the key press Ctrl F6, use:

```
{C_F6}
```

To invoke the original operation of Ctrl F6, use:

```
{'C_F6}
```

`{user macro}` A user macro, which is simply evaluated.

`{statement}` Anything else in braces which doesn't match any of the above expansions, is passed to the script language CScript and executed. Note that it can be a compound statement, and may contain variables, loops etc. Typically, it may be used to introduce a pause into a definition:

```
{pause(20);}
```

or assign a user library function to a key:

```
{userfn();}
```

For clarity, the last semi-colon may be omitted:

```
{pause(20)}
```

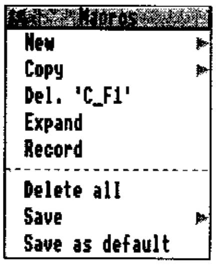
Example

The following example would be used if you wanted to use the terminal with an editor that expects non standard codes for moving the cursor around. The non standard codes are mapped to the cursor keys, and the normal cursor codes are mapped to the shifted cursor keys for use outside the editor. The definition for cursor up would be as follows:

Key/Macro	Definition	Effect
`S_CUp`	`{'CUp}`	defines Shift cursor up to the original key
`CUp`	`|!|[A`	defines cursor up to the new code

Of course, if entering and exiting the editor produces some characteristic string that can be trapped, you could set traps (using the `settrap()` function) to automatically define these macros and remove them afterwards.

Menu options



Click Menu over the **Macros** dialogue box to display the menu shown.

New

This option creates a new macro. If its name corresponds to the name of a key, it will be a Key macro otherwise it will be a User macro.

Copy

This option copies the highlighted macro with the name specified.

Del.

The **Del.** option deletes the highlighted macro.

Expand

The **Expand** option expands the current macro definition.

Record

The **Record** option records all the keyboard presses into the definition of the highlighted macro. It is useful if you wish to program a key or macro with a complicated logon procedure.

First select the macro you wish to use by clicking on it. Now choose the **Record** option, and click back in the terminal window. Do not close the macro window otherwise recording will be stopped. Anything typed into the terminal will be recorded in the macro definition including control codes which are displayed using the standard escape sequences. Switch off recording when you have finished by choosing **Record** or by closing the macro window.

Delete all

Deletes all macros. It cannot be undone, so a warning is given first.

Save keys

This option saves the macros in a script language file. Details of the script command used to define keys, are given in the chapter *CScript*.

Save as default

This saves the macros in the script file `!Macros` in directory `!Hearsay.AutoRun`, which is automatically executed when Hearsay is loaded.

12 CScript

CScript is the embedded script language in Hearsay. This language provides a means of configuring Hearsay for different services, and for executing complicated logon procedures. Advanced users will be able to use it to write complete applications such as host systems. The language syntax is a subset of C with a number of extensions from C++.

CScript programs should be plain ASCII text files, typically written using a text editor such as Edit, and saved to disc. Once saved, they may be executed using one of the methods described below.

Hearsay is compatible with Throwback error handling, supported by many text editors

Running script programs

A CScript script may be executed by doubling-clicking on it from the desktop, or by dragging it to the Hearsay icon on the icon bar. In order to run scripts in this way, they must have the filetype HsyScrip (&D65). After execution, the script and all its functions, variables etc. are discarded. Scripts may call built-in Hearsay functions, or library functions defined in other script files.

A number of example scripts are provided in the Scripts directory on the Hearsay Extras disc. Please read the file `Scripts.ReadMe` for further details.

Library scripts

If you put a script file in the directory `!Hearsay.Library`, then on running Hearsay, any functions or global declarations in that file are added to the resident library of functions. A library function may be called by other CScript scripts, or assigned to a function key and executed when the key is pressed.

The default library file `!Alib` contains a set of standard global declarations used as parameters by many of the built-in functions. This file should not be deleted or modified in any way.

Autorun scripts

If you put a script file in the directory `!Hearsay.AutoRun`, then on running Hearsay, it will be loaded and it's `main()` function executed. After execution, the script and all it's functions, variables etc. are discarded. Typically, auto-run scripts are used to configure Hearsay on start-up. Four auto-run scripts are supplied by default:

- `!Choices` is the script saved by the **Save** option on the **Choices** dialogue box.
- `!Config` is the default configuration script saved by the **Save as default** option on the **Script** menu.
- `!Custom` sets other personal preferences such as file transfer filetype aliases and call band rates.
- !Macros is a set of default key macros.

Number directory

CScript scripts may be executed automatically when a number is dialled from the number directory. You can specify up to 2 scripts that are to be executed before the number is dialled. Typically these are used to configure Hearsay correctly for the service being dialled. You can also specify up to 2 scripts that are to be executed after dialling. These would normally be used for carrying out complicated logon procedures. Please refer to the chapter *Communications* for further details.

Multitasking CScript scripts

In order to make CScript scripts multi-task with other RISC OS programs you have to place the `pause()` function in the main loop of your program. When this function is executed, a WIMP poll occurs. Details of the `pause()` function are given later in the section describing serial input/output functions.

Escaping from programs

If you are polling the WIMP using `pause()`, you can escape from a script at any time by selecting **Stop script** from the **Script** menu. If not, you may terminate programs by pressing Ctrl-Esc.

CScript language details

Comments

The characters /* introduce a comment, which terminates with the characters */. The C++ form // may also be used to introduce a comment which is terminated at the end of the line.

```
/* this is a C style comment
which may extend across any
number of lines */

// this is a C++ style single line comment
```

Identifiers

An identifier is a sequence of letters and digits. The first character must be a letter or underscore _. Upper and lower case identifiers are different. Identifiers may be any length, and all characters are significant.

Keywords

The following identifiers are reserved for use as keywords:

```
break       else        string
case        for         switch
continue    if          void
default     int         while
do          return
```

Integer constants

An integer constant consisting of a sequence of digits is taken to be decimal. A sequence preceded by 0 (zero) is taken to be octal. Octal constants do not contain the digits 8 or 9. A sequence of digits preceded by 0x or 0X (zero, X) is taken to be a hexadecimal integer. The hexadecimal digits include a or A through to f or F representing values 10 through to 15.

For example, decimal 127 can be written as follows:

```
a = 127;  // decimal constant
b = 0177; // octal constant
c = 0x7f; // hexadecimal constant
d = 0X7F; // hexadecimal constant
```

Character constants

A character constant is a sequence of one or more characters enclosed in single quotes, as in `'x'`. The value of the character constant with only one character is the ASCII value of the character. The following escape sequences may be used to represent special characters.

newline	LF	ASCII 10	`\n`
horizontal tab	HT	ASCII 9	`\t`
vertical tab	VT	ASCII 11	`\v`
carriage return	CR	ASCII 13	`\r`
formfeed	FF	ASCII 12	`\f`
double quote	"	ASCII 34	`\"`
backslash	\	ASCII 92	`\\`

String constants

A string constant is a sequence of characters surrounded by double quotes. A null byte is appended to strings so that programs that scan the string can find the end.

In order to represent string constants the character constant escape sequences listed above may be used.

```
s = "ABC";      // set s to string ABC
p = "Hello\r"; // set p to string Hello + CR
t = "\"DEF\""; // set t to string "DEF"
u = "\\";       // set u to string \
```

Variable declarations

Variables may be Global or Automatic. Global variables are declared outside of functions and retain their value throughout the program. Automatic variables may be function parameters or declared inside compound statements, and only exist whilst execution is taking place within their scope.

Global and automatic variables may have the same identifiers.

Three types of variables are supported:

- `void`
- `int` (32-bit signed integers)
- `string` (variable length character strings).

```
int null = 0;     // global declaration

void main(void)
{
  int a, b, c;    // automatic declarations
  int x44 = 99;
  string s;
  string str = "Hearsay II";
  int d = 2;
}
```

Function declarations

Functions can be of type `void`, `int` or `string`, and arguments may be `int` or `string`. Parameters are normally 'call by value', which means that the called function cannot directly alter a variable in the calling function; it can only alter its private, temporary copy. If an ampersand & is placed before a string parameter in a function definition, 'call by reference' is used. In this case the address of the string variable is passed to the called function, which may modify the variable in the calling function. 'Call by reference' may only be used with string parameters.

Like normal C programs, each program must have a function `main()` which is executed first. This implementation does not support function prototypes, so function definitions must appear before they are called in the source file.

```
int length(string s)

{
  int i;
  for (i = 0; schar(s, i) != 0; i++);
  return i;
}

void main(void)
{
  tprinti(length("hello"));
}
```

In the example above, the value that `length` computes is returned to `main` by the `return` statement.

Operators

The rules for precedence and associativity of operators are the same as those for C. However, in this implementation many operators may also be applied to strings.

`l + r` If `l` and `r` are ints, returns the arithmetic sum.
If `l` and `r` are strings, returns result of concatenating `r` to `l`.

`l - r` `l` and `r` must be ints, in which case returns `r` subtracted from `l`.

`l * r` If `l` and `r` are ints, returns the product.
If `l` is a string and `r` an int, returns `l` concatenated to itself, `r` times.
If `r` is a string and `l` an int, returns `r` concatenated to itself, `l` times.

`l / r` If `l` and `r` are ints, returns the quotient.
If `l` and `r` are strings, returns position of `r` in `l`.

`l % r` If `l` and `r` are ints, returns the remainder.
If `l` and `r` are strings, returns `strspn(l, r)`.

`l & r` If `l` and `r` are ints, returns bitwise AND.

`l | r` If `l` and `r` are ints, returns bitwise OR.

`l ^ r` If `l` and `r` are ints, returns bitwise XOR.

`l << r` If `l` and `r` are ints, returns `l` shifted left `r` bits.
If `l` is a string and `r` an int, returns `l` shifted left `r` characters.

`l >> r` If `l` and `r` are ints, returns `l` shifted right `r` bits.
If `l` is a string and `r` an int, returns `l` shifted right `r` characters.

`+l` If `l` is a string, forces it to upper case.

`-l` If `l` is an int, does Unary -.
If `l` is a string, forces it to lower case.

`~l` If `l` is an int, does Ones Complement.
If `l` is a string, swops case.

`!l` Logical NOT. `l` must be an int, in which case TRUE (!=0) becomes FALSE (0), and vice versa.

`l && r` Logical AND. `l` and `r` must be ints. Expression evaluates left to right, and terminates as soon as falsehood is known.

`l || r` Logical OR. `l` and `r` must be ints. Expression evaluates left to right, and terminates as soon as truth is known.

`l , r` Comma operator. l is evaluated then r.

`l ? r : rl` Expression switch. If l is TRUE, returns r else returns rl.

`l == r` Equality operator. l and r must be either ints or strings.

`l != r` Inequality operator. l and r must be either ints or strings.

`l > r` Greater than operator. l and r must be either ints or strings.

`l < r` Less than operator. l and r must be either ints or strings.

`l >= r` Greater or equal operator. l and r must be either ints or strings.

`l <= r` Less or equal operator. l and r must be either ints or strings.

`++l, l++` Increment operators. l must be an int, and an lvalue.

`--l, l--` Decrement operator. l must be an int, and an lvalue.

`l = r` Assignment operator. l and r must be of the same type. l must be an lvalue.

`l += r` Assign l + r to l. l and r must be of the same type. l must be an lvalue.

`l -= r` Assign l - r to l. l and r must be ints. l must be an lvalue.

`l *= r` Assign l * r to l. l and r must be ints. l must be an lvalue.

`l /= r` Assign l / r to l. l and r must be ints. l must be an lvalue.

`l %= r` Assign l % r to l. l and r must be ints. l must be an lvalue.

`l |= r` Assign l | r to l. l and r must be ints. l must be an lvalue.

`l &= r` Assign l & r to l. l and r must be ints. l must be an lvalue.

`l <<= r` Assign l << r to l. l and r must be ints. l must be an lvalue.

`l >>= r` Assign l >>r to l. l and r must be ints. l must be an lvalue.

Statements

In a `while` loop, the statement is executed repeatedly as long as the value of the expression remains true (!= 0). The test occurs before execution of the statement.

`while` (*int expression*) *statement*;

In a `do` loop, the sub-statement is executed repeatedly so long as the value of the expression remains true (!= 0). The test occurs following each iteration.

`do` *statement*; `while` (*int expression*);

In a `for` loop the first expression is evaluated once, and thus initialises the loop. The second expression is evaluated before each iteration, and if it becomes false (0) the loop terminates. The third expression is evaluated after each iteration.

`for` (*initial expression* ; *int expression* ; *loop expression*) *statement*;

In the `if` statement the expression is evaluated, and if it is true (!= 0), the first statement is executed, otherwise the second statement is executed. An `else` is always connected with the last `else`-less `if` at the same block level nesting.

`if` (*integer expression*) *statement*;
`else` *statement*;

The `switch` statement causes control to be transferred to the case statement which matches the value of the expression. There may be one default statement to which control is transferred if no `case` constant matches the expression.

```
switch (int expression)
{
   case integer constant :  statement;
                            break;
   default:                 statement;
                            break;
}
```

The `break` statement causes an immediate exit from a loop.

The `continue` statement causes control to pass to the start of a loop.

The CScript function library

The remainder of this chapter describes the library of Hearsay functions that may be accessed from the CScript language. These functions are split into three groups, and are summarised below.

Group 1: System functions

These functions are concerned only with RISC OS and the CScript language. There are four sets of functions in this group:

RISC OS functions
File input/output functions
String functions
Serial input/output functions

Group 2: Hearsay functions

These functions carry out operations in the Hearsay software, such as downloading a file, dialling a number, dumping a screen etc. There are 7 sets of functions in this group:

Communications functions
General terminal functions
Text terminal functions
Viewdata terminal functions
Tektronix terminal functions
File transfer functions
Printer functions
Spooling functions
Miscellaneous functions

Group 3: Setup functions

The last group of functions simply setup various options and parameters in Hearsay. They allow Hearsay to be configured for different host systems, and personal preferences to be saved. There are 2 sets of functions in this group:

Choices functions
Configuration functions

Conventions

In the function descriptions that follow, some function parameters may take a range of possible arguments. These arguments are normally constants and shown in upper-case, separated with the | character. For example:

```
void setxmodem(int CRC|CHECKSUM);
```

The function `setxmodem()` may be used as follows:

```
setxmodem(CRC);
```

or

```
setxmodem(CHECKSUM);
```

The constants `CRC` and `CHECKSUM` are defined in the default library script `!Alib`.

Other parameters shown in upper-case, but not separated by the | character, control Hearsay options which may be switched on or off. For example:

```
void setcomms(int PREFIX);
```

The value of the parameter `PREFIX` may be 1 to switch it on, 0 to switch it off or -1 for no change. The constants `PREFIX` and `NC` are predefined in the Hearsay library to 1 and -1 respectively, so that:

`setcomms(PREFIX)`	switches the option ON
`setcomms(!PREFIX)`	switches the option OFF
`setcomms(NC)`	no change to the option

RISC OS functions

`int clock(void);`
`clock` returns the time in centi-seconds since Hearsay started.

`int getenvs(string &s);`
`getenvs` gets the value of the environment variable specified by `s`. If the variable exists it returns 1 with the value of the variable in `s`, otherwise it returns 0.

`void fx(int a, int x, int y);`
`fx` performs an os_byte call.

`int osclis(string &s);`
`osclis` passes the string `s` to the command line interpreter to execute as a command. It returns 0 if the command is successful, otherwise 1 with an error message in `s`.

`int systems(string &s);`
`systems` passes the string `s` to the command line interpreter to execute as a command. If `s` is prefixed by the string `call:`, the calling application is copied to the top of memory first. When the called application terminates, the calling application is restored. It returns 0 if the command is successful, otherwise -2 for failure.

`void errorbox(string &message);`
`errorbox` raises an error, displays a wimp error box containing message, and terminates the program.

`void exit(int status);`
`exit` terminates script execution. The argument `status` is included only for compatibility with C.

`void bbc_vdu(int c);`
`bbc_vdu` outputs the single character `c`.

`int bbc_get(void);`
`bbc_get` waits for a key to be pressed, and returns its ASCII value.

`int bbc_adval(int n);`
If n > 0, `bbc_adval` returns data from the analogue port. If `n` < 0, `bbc_adval` returns information about various buffers.

`int osversion(void);`
`osversion` returns the version of RISC OS in use.

`int bbc_inkey(int c);`
If c > 0, `bbc_inkey` waits for the specified time for a key to be pressed, and returns the ASCII value of the key, or -1 if no key pressed. If c < 0, `bbc_inkey` returns -1 if the specified key is being pressed at that instant, otherwise it returns 0.

`int swil3(int swi, int r0, int rl, int r2);`
`swil3` performs the SWI instruction specified by `swi`, setting registers `r0`, `rl`, and `r2`. It returns the contents of register `rl`.

`int objectexists(string &name);`
`objectexists` returns 1 if `name` is a file, 2 if `name` is a directory, or 0 if it does not exist.

`void startscan(void);`
`startscan` initialises the function `nextobject`.

`int nextobject(string &dir, string &wildcard, string &name);`
`nextobject` searches directory `dir` for the next object which matches the wildcarded name `wildcard`, and returns its name in the string `name`. It returns 1 if the object found is a file, 2 if it is a directory, or 0 if there are no more objects. This function should be initialised once before being used, using `startscan`. The following example prints all the objects in directory $ whose names start with begin with `!`.

```
void main(void)
{
  string s;

  startscan();
  while(nextobject("$", "!*", s))
    tprints(s);
}
```

`void printi(int i);`
`printi` prints the integer `i` to the vdu.

`void prints(string &s);`
`prints` prints the string `s` to the vdu.

`int confirm(string &s);`
`confirm` displays the standard Hearsay confirmation box containing Yes and No icons, and the message s. It returns 1 for YES, 0 for NO or -1 if the user presses Esc or clicks on the background.

File input/output functions

`int fileopen(string &name, string &mode);`
`fileopen` opens the file `name` and returns a handle associated with the file, otherwise 0 if the operation fails. The string `mode` should be one of the following sequences:

”r”	open text file for reading
”w”	create text file for writing
”a”	append; open or create text file for writing at end of file
”r+”	open text file for update (i.e. reading and writing)
”w+”	create text file for update
”a+”	append; open or create text file for update, writing at end

If `mode` includes `b` after the initial letter, as in `rb`, it indicates a binary file.

`int fileclose(int handle);`
`fileclose` closes the file `handle`, or returns non-zero if the operation fails.

`int filewrites(string &s, int handle);`
`filewrites` writes string `s` to file `handle`, or returns non-zero if an error occurs.

`int filewritei(int i, int handle);`
`filewritei` writes int `i` to file `handle`. It returns non-zero if an errors occurs.

`int fileerror(int handle);`
`fileerror` returns non-zero if the error indicator for the file `handle` is set.

`int fileeof(int handle);`
`fileeof` returns non-zero if the end-of-file indicator for the file `handle` is set.

`int filereadi(int handle);`
`filereadi` returns the next int from the file `handle`, or -1 if end-of-file or an error occurs.

`int filereads(string &s, int handle);`
`filereads` reads the next string terminated by a newline from the file `handle` into string `s`. The newline character is included in the string, which is terminated by a null character. It returns non-zero if end-of-file or an error occurs.

`int fileputc(int c, int handle);`
`fileputc` writes character `c` (converted to an int) to file `handle`. It returns the character written, or -1 if an error occurs.

`int filegetc(int handle);`
`filegetc` returns the next character of the file `handle` (converted to an int), or -1 if end-of-file or an error occurs.

`int fileseek(int offset, int handle);`
`fileseek` sets the file position for the file `handle`, to the position `offset` characters from the start of the file. It returns non-zero on an error.

`int filetell(int handle);`
`filetell` returns the current file position for the file `handle`, or -1 on an error.

String functions

```
int stoi(string &s);
```
stoi returns the string s converted to an integer.

```
string itos(int i);
```
itos returns the integer i converted to a string.

```
string itoxs(int i);
```
itoxs returns the integer i converted to a hexadecimal string.

```
int schar(string &s, int n);
```
schar returns the ASCII code for character n of string s.

```
string chars(int c);
```
chars returns a string consisting of the character equivalent of the ASCII code c.

```
int slen(string &s);
```
slen returns the length of the string s.

```
string mids(string &s, int from, int len);
```
mids returns a segment of string s, commencing at character from and len characters in length.

Serial input/output functions

`void sprints(string &s);`
`sprints` sends the string `s` to the serial port

`void sprinti(int i);`
`sprinti` sends the int `i` (converted to a string), to the serial port.

`void tprints(string &s);`
`tprints` prints the string `s` to the current terminal.

`void tprinti(int i);`
`tprinti` prints the int `i` (converted to a string) to the current terminal.

`int sgetc(int time);`
`sgetc` waits a maximum of `time` centi-seconds for a character (converted to an int) from the serial port. It returns the character read, or -1 on failure.

`int sputc(int c);`
`sputc` writes the character `c` (converted to an int) to the serial port. It returns non-zero on failure.

`void tputc(int c);`
`tputc` writes the character `c` (converted to an int) to the current terminal.

`void kprints(string &s);`
`kprints` sends the string `s` to the serial port using the keyboard channel i.e. as if you had typed it. This causes macro expansion to be carried out, so you can include things like `{CLeft}` in the string. Please refer to the chapter *Macros* for details of the Hearsay macro facilities.

`int kgetc(int time);`
`kgetc` waits `time` centi-seconds or until a byte is received from the terminal output buffer. It returns the byte or -1 if no byte received. To use this function, you must call `claimkeyboard()` first.

`void claimkeyboard(int claim);`
`claimkeyboard` allows you to capture all terminal serial output i.e. usually what is typed. The parameter `claim` should be set to 1 to claim output.

`int getprompt(string &prompt, int time);`
`getprompt` waits a maximum of `time` centi-seconds for the string `prompt` to be received from the serial port. It returns 1 on success or 0 on failure. Incoming data is passed on to the terminal whilst this function executes.

`int sreads(string &s, int timeout, int max);`
`sreads` tries to read a string terminated by a CR or LF from the serial port, of `max` characters in length, within `timeout` centi-seconds. Data is not passed on. The string is terminated by a CR or LF. It returns 1 on success, or 0 on failure.

`int sreadtext(string &s, int timeout, int max);`
`sreadtext` tries to read the string `s` of `max` characters from the serial port. It returns 1 if `s` was terminated with CR/LF. No control codes are passed on and the string is terminated with 0.

```
void settrap(int TX|RX, int type, int add,
string &function, string &trap);
```

`settrap` calls a user defined function when the string `trap` is detected in a serial channel. The parameter `add` should be 1 to set a trap, or 0 to remove it. The parameter `function` is the name of the function to be called, and MUST be of the form:

```
int trapfunction(string &trap);
```

The parameter type should be 0 or 1.

0 Event trap. In this type of trap, all data received is passed on to the terminal. Here we just want to know that the string `DISCONNECT`, say, was received. We do not want to stop everything when the letter `D` appears.

1 Filter trap. In this type of trap, data is not passed on whilst the match is established. It may be used to prevent a control sequence from being sent or received. Notice that you can change the trap string, so you can replace one string in the serial stream with another.

The following example shows how the word `color` is replaced by `colour` using a type 1 trap.

```
settrap(RX, 1, 1, "fix", "color");

int fix(string &trap)
{
  trap = "colour";
  return 1;
}
```

If the trap function returns 1, the trap string is passed on, otherwise it won't be. Normally, a type 1 trap function will return 1 and a type 0 trap function will return 0. This is because type 0 traps have already passed on most of the trap string when the trap goes off.

Communications functions

When communicating with the modem, you should where possible use the functions described below rather than calling the low-level modem driver functions directly (the low-level functions are described in *Appendix F*). Doing this will ensure that the main program keeps track of events. Please note that the functions below are only available if the modem and modem driver you are using, supports them.

`void connect(void);`
`connect` attempts puts the modem online.

`void disconnect(void);`
`disconnect` attempts to put the modem offline.

`void talktomodem(void);`
`talktomodem` temporarily disconnects the line, allowing you to send commands to the modem.

`void reconnect(void);`
`reconnect` connects back to the line after using `talktomodem()`.

`void dial(string &number);`
`dial` dials the number specified (converted to a string).

`void autoanswer(int rings);`
`autoanswer` answers the phone and connects the line after the specified number of `rings`.

General terminal functions

```
void setterminal(VIEWDATA|MINITEL|ANSI|VT320|
VT102|VT52|TEK4105|TELETYPE|CAMPUS);
```

setterminal opens the specified terminal. It is just like opening a terminal from the Hearsay icon bar menu.

```
int termchar(int VIEWDATA|MINITEL|ANSI|VT320|
VT102|VT52|TEK4105|TELETYPE, int x, int y);
```

termchar returns the character at position x, y in the terminal window.

```
int termcurs(int VIEWDATA|MINITEL|ANSI|VT320|
VT102|VT52|TEK4105|TELETYPE, int coord);
```

termcurs returns the coordinates of the cursor in the specified terminal. If the parameter coord is set to 0 it returns the X coordinate, or if coord is set to 1 it returns the Y coordinate.

```
void termline(int VIEWDATA|MINITEL|ANSI|VT320|
VT102|VT52|TEK4105|TELETYPE|CAMPUS|NOTERM);
```

termline controls which terminal reads the serial line. The parameter NOTERM stops any terminal reading the line. The parameter set here is reflected in the **Line** setting on the **Line mode** menu.

```
void termtab(int VIEWDATA|MINITEL|ANSI|VT320|
VT102|VT52|TEK4105|TELETYPE|CAMPUS, int x,
int y);
```

termtab tabs the cursor in the terminal specified, to the given position. The x and y coordinates are compatible with those returned by termcurs().

```
void setband(int band, int secs1, int secs2,
int secs3);
```

setband sets up a call band entry. The parameter band should be a value between 1 and 5 (relating to L, a, b1, b and m). The parameters secs1, secs2 and secs3 are the number of seconds per unit for each time band (relating to cheap, standard and peak).

```
void setcurband(int band);
```

setcurband sets the band for the purpose on the clock on the status line or keypad. If band == -1 the real time is displayed, if band == 0 the time on line is displayed, or if band > 0 the normal display is used.

`int savescreen(string &filename, int VIEWDATA|MINITEL|ANSI|VT320|VT102|VT52|TEK4105|TELETYPE, int CEPT3|CEPT2|SPRITE| DRAW|TEXT|RAWDATA);`
`savescreen` saves the specified terminal screen in the file named `filename` with the specified `filetype`. It returns 0 on failure. The possible filetypes are as follows:

CEPT3	D33	Viewdata terminal only
CEPT2	D32	Minitel terminal only
SPRITE	FF9	Viewdata terminal only
DRAW	AFF	Tektronix terminal only
TEXT	FFF	VT, Viewdata and Minitel terminals only
RAWDATA	FF1	VT terminal only

`void setcampusdefault(VIEWDATA|VT102);`
`setcampusdefault` controls which terminal Campus 2000 starts up in. This command is used in `!Hearsay.AutoRun.!Custom` to set the default terminal to `VIEWDATA`.

Text terminal functions

```
void vtcharmap(int USASCII|BRITISH|DUTCH|
FINNISH|FRENCH|FRENCHCAN|GERMAN|ITALIAN|
NORWEGIAN|PORTUGUESE|SPANISH|SWEDISH|SWISS|
LATIN1|SPECGRAPH|SUPPGRAPH|TECHNICAL|ALTROM|
ALTGRAPH|DRCS, int charnumber,
int spritenumber);
```

`vtcharmap` allows the mapping of the VT character sets to be altered. The parameter `charnumber` is the ASCII character number between 0 and 95. The parameter `spritenumber` is the number of the character in the sprites files.

There are three sprite files used by the VT terminal located in the `!Hearsay.Resources` directory:

VT = sprites 0 to 255
ANSI = sprites 256 to 511
VTX = sprites 512 to 640.

High resolution versions of these characters are also available in files of the same name but with the suffix `18`. These characters are used in multisync screen modes.

```
void setvtcolour(int colour, int type,
int wimpcol, int r, int g, int b);
```

`setvtcolour` is used to set which colours the VT emulator will use. The parameter colour should be one of the following values:

0	BLACK
1	RED
2	GREEN
3	YELLOW
4	BLUE
5	MAGENTA
6	CYAN
7	WHITE

The parameter type should be one of the following values:

0	NORMAL
1	BOLD
2	FAINT

The parameter `wimpcol` is the 16-colour mode wimp colour, and parameters `r`, `g` and `b` are the 256-colour mode components.

`void setvtident(int VT320|VT102|VT52,`
`string &ident);`
`setvtident` sets the identifier response for the specified terminal. The parameter `ident` should be any string less than 32 characters long.

`void setvtdevatt(string &attributes);`
`setvtdevatt` sets the secondary device attributes response. The parameter `attributes` should be any string less than 32 characters long.

The following functions are script language equivalents of the options on the **Select** menu in the text terminals.

`void vtselectprint(void);`
`vtselectprint` prints the selection.

`void vtselectsend(void);`
`vtselectsend` transmits the selection.

`void vtselectspool(void);`
`vtselectspool` appends the selection to the open spool file.

`void vtselectcopy(void);`
`vtselectcopy` make a copy of the selection.

`void vtselectpaste(void);`
`vtselectpaste` transmits the copied selection.

The following example shows how `vtselectprint()` can be used to implement the keyboard short cut Ctrl-V to print the selected text:

```
    defmacro("C_V", "{vtselectprint()}");
```

The above definition could be located in an autorun script, or alternatively the macro could be defined in the macros dialogue box.

Viewdata terminal functions

```
void setvxcolour(int colour, int wimpcol, int r,
int g, int b);
```

`setvxcolour` is like setvtcolour but for the Viewdata terminal.

```
void setvxkeypadcode(int VIEWDATA|MINITEL,
int button, string &s);
```

`setvxkeypadcode` sets the codes that are sent when a Viewdata or Minitel keypad button is clicked. The button numbers are as follows:

Viewdata		**Minitel**	
Index	0	Envoi	0
Retran	1	Suite	1
Back	2	Retour	2
Leave	3	Guide	3
		Repttn	4
		Sommre	5
		Annltn	6
		Corrct	7
		Fin	8

```
void setvxkeypadstring(int VIEWDATA|MINITEL,
int button, string &s);
```

`setvxkeypadstring` sets the `string` written in the button icon.

```
void setvxlanguage(int VIEWDATA|MINITEL,
int language);
```

`setvxlanguage` writes the appropriate strings for a given language to the keypad icons. Currently supported values for language are BRITISH and FRENCH

```
void setvxterminator(int VIEWDATA|MINITEL,
string &s);
```

`setvxterminator` allows you to define the string that is sent when Return is pressed. The normal value is _ (which is translated into # in the Viewdata terminal).

The strings in the four functions above are limited to 16 chars long.

```
int vxbuffopen(string &filename);
```
`vxbuffopen` opens a Viewdata buffer with the name specified by `filename`. If `filename` already exists, then that buffer will be loaded. This function returns a handle on the buffer, or 0 if the operation fails.

```
int vxbuffseek(int handle, int frame);
```
`vxbuffseek` moves to the frame specified by the parameter `frame`, in the buffer specified by `handle`, and returns 0 on failure.

```
int vxbuffop(int handle, int opcode);
```
`vxbuffop` carries out various operations on the buffer specified by handle. It returns 1 on success, or 0 on failure. The parameter `opcode` should specify one of the following operations:

0	tag terminal frame to end of buffer
1	insert terminal frame into buffer
2	replace buffer frame with terminal frame
3	delete current frame in buffer
4	paste current frame into Viewdata terminal
5	go to current frame

```
int vxbuffclose(int handle);
```
`vxbuffclose` closes the buffer specified by `handle`, saving the buffer to disc if it has been modified. It returns 1 on success, or 0 on failure,

```
void vxbuffshow(int handle, int show);
```
`vxbuffshow` opens or closes a window on the buffer specified by `handle`. The parameter `show` should be set to 0 to close the window, or 1 to open the window.

```
void sendframe(void);
```
`sendframe` sends the current Viewdata frame. It is equivalent to the **Send frame** option on the Viewdata **Action** menu.

Tektronix terminal functions

```
void settekversion(int version);
```

`settekversion` defines the response made by Hearsay when requested the version of the Tektronix terminal.

```
void settekfont(string &fontname,
int fontwidth);
```

`settekfont` allows you to specify the font name that the Tektronix terminal will use in the Draw files it produces. The parameter `width` is the fixed width of all the characters in the font, in 1/1000 em.

File transfer functions

`void sendfiles(void);`
`sendfiles` sends the files in the TX batch.

`void receivefiles(void);`
`receivefiles` receives files to the RX batch.

`void addrxalias(int file_type,`
`string &extension);`
`addrxalias` adds a new RX alias. Files received whose extension matches, will be given the defined type.

`void addtxalias(int file_type,`
`string &extension);`
`addtxalias` adds a new TX alias. Files sent with the given type will have the given extension added.

`void setdefltalias(int file_type,`
`string &extension);`
`setdefltalias` sets the file type and extension that will be used by default.

`void emptybatch(int TX|RX);`
`emptybatch` empties files from the specified batch. The first parameter specifies the batch, and should be set to TX or RX.

`int addtobatch(string &local_filename);`
`addtobatch` adds the file whose name is `local_filename`, to the TX batch. It returns the number of the file in the batch, or 0 on failure. Please note that the files in the batch are in alphabetical order, so after a file has been added, the numbers of the other files may change.

`void batchremotename(int TX|RX, int filenumber,`
`string &remotename);`
`batchremotename` sets the remote name of the file specified by `filenumber`, to the name `remotename`. The first parameter specifies which batch the file is in.

`void batchsetstate(int TX|RX, int filenumber,`
`int flags);`
`batchsetstate` sets the sent/ready state of the file specified by `filenumber`, in the batch specified. The parameter `flags` should be set to 1 to indicate that the file has been sent, or 3 to indicate that it is ready.

```
int batchgetstate(in TX|RX, int filenumber);
```

batchgetstate returns the state of the file specified by filenumber, in the batch specified. The values returned are:

0	no state set	3	file ready
1	file sent	4	file open
2	file short	5	file received

```
int batchgetname(int TX|RX, int n, int local,
string &name);
```

batchgetname returns the name of the file at position n in the batch specified. The position n should start from 1. The name is returned in the string name. If the parameter local is non-zero the name returned will be the local filename, otherwise the remote filename. This function returns 0 if file n does not exist.

```
void batchremove(int TX|RX, int filenumber);
```

batchremove removes the specified file from the TX or RX batch.

```
void kermitserver(void);
```

kermitserver causes Hearsay to go into Kermit server mode.

```
void kermitget(string &cmdline);
```

kermitget issues the Kermit Get command.

```
void kermitbye(void);
```

kermitbye issues the Kermit Bye command.

```
void kermitfinish(void);
```

kermitfinish issues the Kermit Finish command.

```
void kermitremdir(string &cmdline);
```

kermitremdir issues the Kermit Remote dir. command.

```
void kermitchangeremdir(string &cmdline);
```

kermitchangeremdir issues the Kermit Remote CWD command.

```
void kermittype(string &cmdline);
```

kermittype issues the Kermit Remote type command.

```
void kermitdelete(string &cmdline);
```

kermitdelete issues the Kermit Remote delete command.

```
void kermitremote(string &cmdline);
```

kermitremote issues a Kermit Remote host command.

```
void setftpbuffersize(int txsize, int rxsize);
```

setftpbuffersize sets the size of the RAM buffers used for received or transmitted files. It is set to 4K by default.

Printer functions

```
void vtautoprint(int state);
```

`vtautoprint` switches on or off the auto print mode in the VT terminal. The parameter `state` should be set to 1 to switch auto print on, or 0 to switch it off.

```
void tekdump(void);
```

`tekdump` does a dump of the Tektronix terminal screen. The type of dump carried out depends on the `settekprint()` function.

```
void vxdump(void);
```

`vxdump` does a dump of the Viewdata terminal screen. The type of dump carried out depends on the `setvxprint()` function.

```
void vtdump(void);
```

`vtdump` does a dump of the VT terminal screen. The type of dump carried out depends on the `setvtprint()` function.

Spooling functions

```
int spoolopen(string &filename, int TEXT|
RAWDATA|CEPT3|CEPT2);
```

`spoolopen` opens a spool file with the name `filename`, and returns 1 on success. The last parameter determines the type of spool file to be opened.

```
void spoolclose(void);
```

`spoolclose` closes the currently open spool file.

```
void spool(int state);
```

`spool` allows spooling to be switched on or off. The parameter `state` should be set to 0 to switch spooling off, or 1 to switch it on.

Miscellaneous functions

`void vasscomnegotiate(void);`
`vasscomnegotiate` makes the Vasscom error correction protocol negotiate a link. This sets off a background process; Vasscom is not necessarily on or off when this function returns.

`void terminate(void);`
`terminate` closes down Hearsay and quits it.

`void setmenusense(int REVERSE);`
Normally, pressing the menu button displays the main menu, and holding down Ctrl and pressing Menu displays the User menu. If `setmenusense` is called with parameter REVERSE, the operation is reversed. If no user menu macros are defined, the main menu is always displayed. To change back to normal operation, the function should be called with the parameter 0.

`void pause(int time);`
`pause` causes a delay of `time` centi-seconds. WIMP polling will continue in this function, so it should be used in the main loop of your program if you want the rest of the programs on the desktop to multi-task.

`void translatetable(int TX|RX, int oldcode,`
`int newcode);`
`translatetable` sets up a character translation table. It causes the character `oldcode`, to be translated to the character `newcode` (both characters converted to ints). The first parameter determines the stream. Please note that these translation tables are also used by the ASCII file transfer protocol.

`void cleartranslatetable(int TX|RX);`
`cleartranslatetable` clears the character translation table for the stream specified.

`void setpoll(int on);`
`setpoll` can be used to switch polling on and off. Typically it may be used during modem initiation and termination when you don't want polling.

`void setstate(int ACTIVE);`
`setstate` is used to control whether Hearsay is active or not. Please refer to *Appendix G* for further details.

```
void delallmacros(void);
```

`delallmacros` deletes all macros.

```
void delmacro(string &name);
```

`delmacro` deletes the macro called name.

```
void defmacro(string &name, string &defn);
```

defmacro defines the macro called `name` with the definition `defn`.

Choices functions

The following functions are used in the `AutoRun.!Choices` file, and are saved whenever you use the **Save choices** option on the **Choices** dialogue box. They simply setup a number of Hearsay defaults, so it is unlikely that you will need to use these in programs.

```
void specifydriver(string &modem_driver_name);
```

`specifydriver` loads a new modem driver. Please note that the new driver is not loaded until after the current script has terminated. The modem drivers available are saved in directory `!Hearsay.Driver`.

```
void setdefterminal(int VIEWDATA|MINITEL|
ANSI|VT320|VT102|VT52|TEK4105|TELETYPE|CAMPUS);
```

`setdefterminal` sets the default terminal which is opened when you click on the Hearsay icon on the icon bar.

```
void setchoices(int NEWDATA, int CONFIRM,
int LOGFILE, int DRIVER|TEXTPRINT|EPSON|IBM,
int TONE);
```

`setchoices` switches various choices on and off.

```
void setprinter(int height, int width,
int CR|LF|CRLF|LFCR|RAW);
```

`setprinter` sets the number of lines and columns on a page which are to be used when the RISC OS printer drivers are used for continuous text printing. The final parameter is the end-of-line character sent to the printer. If RAW is used the EOL character is passed on unchanged, otherwise CR characters are ignored and newlines are mapped to the character specified.

Configuration functions

The following functions may be used to configure all aspects of Hearsay. The options in all the Hearsay dialogue boxes and menus may be set using these functions. Whenever you use the **Save script** or **Save as default** options, a complete set of these functions is saved to file.

```
void setspeed(int txrate, int rxrate);
```
`setspeed` sets the TX and RX baud rates.

```
void setbits(int data, int N|E|O|M|S, int stop);
```
`setbits` sets the data format.

```
void setline(int RTSCTS|NONE|XONXOFF, FILTER,
ANSWER);
```
`setline` sets the general line settings.

```
void setlink(int NONE|VASSCOM|MNP, TRICKLE);
```
`setlink` sets the link level protocol.

```
void setvasscom(int TXBLOCK, int RXBLOCK,
int blocksize);
```
`setvasscom` configures the Vasscom link level protocol.

```
void setftp(int XMODEM|XMODEM1K|YMODEM|
ZMODEM|KERMIT|SEALINK|ASCII|CET);
```
`setftp` sets the default file transfer protocol.

```
void setxmodem(int CRC|CHECKSUM);
```
`setxmodem` configures the Xmodem file transfer protocol.

```
void setymodem(int CRC|CHECKSUM, 128|1024);
```
`setymodem` configures the Ymodem file transfer protocol.

```
void setzmodem(int CRC32, int AUTORESUME,
int AUTODLOAD);
```
`setzmodem` configures the Zmodem file transfer protocol.

```
void setcet(int SLOW, int CET, string &eol,
int AUTORESUME);
```
`setcet` configures the CET file transfer protocol. The parameter `eol` specifies the end-of-line character, and may use the `|` syntax e.g. `|M`.

```
void setkermit(int BINARY, int WINDOWS,
int QUOTE8, string &q8char, int CHECK1|CHECK2|
CHECK3);
```
`setkermit` configures the Kermit file transfer protocol.

```
void setkermitparam(int TX|RX, int packetsize,
int npadchars, string &padchar,
string &startchar, string &eolchar, int timeout,
string &ctrlqchar);
```

`setkermitparam` configures the Kermit TX and RX parameters.

```
void setasciitx(string &txpromptchar, int txeol,
string &txbeginche, string &txendchar,
string &txstartchar, string &txstopchar,
int txcharpace, int txlinepace, int EXPAND,
int LOCAL);
```

`setasciitx` configures the ASCII file transfer protocol TX parameters.

```
void setasciirx(string &rxpromptchar, int rxeol,
string &rxtimepromptchar, int rxtimeout,
string &rxbeginchar, string &rxendchar,
int LOCAL);
```

`setasciirx` configures the ASCII file transfer protocol RX parameters.

```
void setbatch(int LARGE|SMALL|INFO, int
NAME|TYPE|SIZE|DATE);
```

`setbatch` controls the display format used by the batch filers.

```
void setbatchconfig(int REMOVE, int SEND,
int ALIAS, int RETAIN, int PROMPT, int WARN,
int OVERWRITE, int DISCARD);
```

`setbatchconfig` sets the batch configuration options.

```
void setvtfile(int CTRLCODES, int COMPLEMENT);
```

`setvtfile` sets the VT terminal file options.

```
void setvtprint(int REMOTE, int FF, int FULL);
```

`setvtprint` sets the VT terminal print options.

```
void setvtdisplay(int REVERSE, int INSERT,
int WRAP, int SMOOTH, int 80|132, int CURSOR,
int BLOCK, int DESTBKSPC, int NONE|TABS|LOCAL|
HOST, int mul, int div, int VARIABLE);
```

`setvtdisplay` sets the VT terminal display options.

```
void setvtset(int G0|G1|G2|G3,int USASCII|
BRITISH|DUTCH| FINNISH|FRENCH|FRENCHCAN|GERMAN|
ITALIAN|NORWEGIAN|PORTUGUESE| SPANISH|SWEDISH|
SWISS|LATIN1|SPECGRAPH|SUPPGRAPH|TECHNICAL|
ALTROM|ALTGRAPH|DRCS);
```

`setvtset` sets the VT terminal character sets.

```
void setvtgrl(int GL|GR,int G0|G1|G2|G3);
```

`setvtgrl` sets the GL and GR character sets.

```
void setvtkey(int NEWLINE, int REPEAT,
int CURSOR, int KEYPAD, int DELBKSPC);
```

`setvtkey` sets the VT terminal keyboard options.

```
void setvtgeneral(string &answerback, int BELL,
int buffersize, int LOCK);
```

`setvtgeneral` sets the VT terminal general options.

```
void setvtline(int LOCAL, int ECHO, int TXCRLF,
int RXCRLF, int ACT|IGNORE|DISPLAY, int BIT8);
```

`setvtline` sets the VT terminal line options.

```
void setvttabs(string &tabs);
```

`setvttabs` sets the VT terminal tab stops. The string parameter tabs is a string of the form `.....T....` up to 132 characters long. The T's denote VT tab positions.

```
void setvxfile(int CTRLCODES, int COMPLEMENT,
int CEPT3);
```

`setvxfile` sets the Viewdata terminal file options.

```
void setvxdisplay(int KEYPAD, int CURSOR,
int BLOCK, int mul, int div, int VARIABLE);
```

`setvxdisplay` sets the Viewdata terminal display options.

```
void setvxprint(int TEXTDUMP|GRPOS|GRNEG,
int PORTRAIT, int xscale, int yscale, int cx,
int cy, int FF);
```

`setvxprint` sets the Viewdata terminal print options. The parameters `xscale` and `yscale` should be percentages. The parameters `cx` and `cy` should be in millimetres.

```
void setvxkeyboard(int RETURN, int REPEAT);
```

`setvxkeyboard` sets the Viewdata terminal keyboard options.

```
void setvxgeneral(string &answerback, int BELL,
string &mbxreply);
```

`setvxgeneral` sets the Viewdata terminal general options.

```
void setvxsend(string &prefix, string &suffix,
string &asterixseq, string &atseq,
string &solseq, int SHORT, int ECHO,
int echopace, int ESCSEQ);
```

`setvxsend` sets the Viewdata terminal frame send options.

```
void setvxline(int LOCAL, int ECHO, int TXCRLF,
int RXCRLF, int ACT|IGNORE|DISPLAY, int BIT8);
```

`setvxline` sets the Viewdata terminal line options.

```
void setvxbuff(int TOOLS, int mul, int div,
int VARIABLE);
```

`setvxbf` sets the Viewdata terminal tab buffer options.

```
void setcampuskeyboard(int RETURN);
```

`campuskeyboard` controls whether Return sends # in the Campus 2000 terminal.

```
void settekfile(int CTRLCODES, int COMPLEMENT);
```

`settekfile` sets the Tektronix terminal file options.

```
void settekprint(int GRPOS|GRNEG, int REMOTE,
int MARGINS,int PORTRAIT, int xscale,
int yscale, int cx, int cy, int copies);
```

`settekprint` sets the Tektronix terminal print options. The parameters `xscale` and `yscale` should be percentages. The parameters `cx` and `cy` should be in millimetres.

```
void settekgeneral(int TEK4010, int TEXT,
int LOCK, int mul, int div, int VARIABLE);
```

`settekgeneral` sets the Tektronix terminal general options.

```
void settekindex(int index, int r, int g,
int b);
```

`settekindex` sets the Tektronix terminal colour index. The parameter `index` should be a value between 0 and 15. The parameters `r`, `g` and `b` should be values between 0 and 255.

```
void settekline(int index);
```

`settekline` sets the Tektronix colour `index` which will be used for lines.

```
void settektext(int index);
```

`settektext` sets the Tektronix colour `index` which will be used for text.

```
void settekfill(int index);
```

`settekfill` sets the Tektronix colour `index` which will be used for filling.

```
void settekback(int index);
```

`settekback` sets the Tektronix colour `index` which will be used for the background.

```
void setautoredial(int REDIAL, int attempts,
int delay);
```

`setautoredial` sets the auto redial options.

```
void setcomms(int PREFIX);
```

`setcomms` sets whether the dial prefix should be used for dialling.

Software vectors

At certain events, Hearsay will attempt to call the following functions:

`void sys_terminate(void);`
`sys_terminate` is called just before termination of Hearsay.

`void sys_term_create(int term);`
`sys_term_create` is called after a terminal is created. The parameter `term` is the terminal type.

`void sys_term_open(int term);`
`sys_term_open` is called after a terminal is opened. The parameter `term` is the terminal type.

`void sys_term_close(int term);`
`sys_term_close` is called after a terminal is closed. The parameter `term` is the terminal type.

`void sys_term_destroy(int term);`
`sys_term_destroy` is called before a terminal is destroyed i.e. when opening a new terminal or using the **Min memory** option. The parameter `term` is the terminal type.

`void sys_ftp_start(int TX|RX, int ftp);`
`sys_ftp_start` is called after a file transfer starts. The first parameter will be -1 when the Kermit server starts and ends.

`void sys_ftp_end(int TX|RX, int ftp);`
`sys_ftp_end` is called after a file transfer ends. The first parameter indicates whether a file is being sent or received. The parameter `ftp` specifies the file transfer protocol. The first parameter will be -1 when the Kermit server starts and ends.

`void sys_ftp_fileopen(string &name, int TX|RX);`
`sys_ftp_fileopen` is called when a file transfer opens a file. The parameter name is the name of the file, and the second parameter indicates whether a file is being sent or received.

`void sys_ftp_fileclose(string &name, int state);`
`sys_ftp_fileclose` is called when a file transfer closes a file. The parameter name is the name of the file, and the parameter `state` is the sent/ready state of the file (see function `batchgetstate()`).

`void sys_online(string &systemname, int online);`
`sys_online` is called when Hearsay goes online or offline. The parameter `systemname` is the service that has just been dialled.

Appendix A

Control codes

The following control codes are supported by the text terminals.

Dec	Hex	Ctrl code	Token	Function
0	&00	|@	NUL	Ignored
1	&01	|A	SOH	Ignored
2	&02	|B	STX	Ignored
3	&03	|C	ETX	Ignored
4	&04	|D	EOT	Ignored
5	&05	|E	ENQ	Transmit answerback message
6	&06	|F	ACK	Ignored
7	&07	|G	BEL	Sound beep
8	&08	|H	BS	Backspace
9	&09	|I	HT	Horizontal tab
10	&0A	|J	LF	Linefeed
11	&0B	|K	VT	Vertical tab, treated as LF
12	&0C	|L	FF	Formfeed, treated as LF or CLS
13	&0D	|M	CR	Carriage return
14	&0E	|N	SO	Map G1 set to GL (lock shift)
15	&0F	|O	SI	Map G0 set to GL (lock shift)
16	&10	|P	DLE	Ignored
17	&11	|Q	DC1	Send XON char (resume)
18	&12	|R	DC2	Ignored
19	&13	|S	DC3	Send XOFF char (suspend)
20	&14	|T	DC4	Ignored
21	&15	|U	NAK	Ignored
22	&16	|V	SYN	Ignored
23	&17	|W	ETB	Ignored
24	&18	|X	CAN	Cancel current escape sequence
25	&19	|Y	EM	Ignored
26	&1A	|Z	SUB	Treated as CAN
27	&1B	|[	ESC	Start escape sequence
28	&1C	|\	FS	Ignored
29	&1D	|]	GS	Ignored
30	&1E	|^	RS	Ignored
31	&1F	|_	US	Ignored

<table>
<tr><th>Dec</th><th>Hex</th><th>Ctrl code</th><th>Token</th><th>Function</th></tr>
<tr><td>128</td><td>&80</td><td>|!|@</td><td></td><td>Ignored</td></tr>
<tr><td>129</td><td>&81</td><td>|!|A</td><td></td><td>Ignored</td></tr>
<tr><td>130</td><td>&82</td><td>|!|B</td><td></td><td>Ignored</td></tr>
<tr><td>131</td><td>&83</td><td>|!|C</td><td></td><td>Ignored</td></tr>
<tr><td>132</td><td>&84</td><td>|!|D</td><td>IND</td><td>Index, move cursor down one line</td></tr>
<tr><td>133</td><td>&85</td><td>|!|E</td><td>NEL</td><td>Next line, treated like CR/LF</td></tr>
<tr><td>134</td><td>&86</td><td>|!|F</td><td>SSA</td><td>Ignored</td></tr>
<tr><td>135</td><td>&87</td><td>|!|G</td><td>ESA</td><td>Ignored</td></tr>
<tr><td>136</td><td>&88</td><td>|!|H</td><td>HTS</td><td>Set horizontal tab at cursor</td></tr>
<tr><td>137</td><td>&89</td><td>|!|I</td><td>HTJ</td><td>Ignored</td></tr>
<tr><td>138</td><td>&8A</td><td>|!|J</td><td>VTS</td><td>Ignored</td></tr>
<tr><td>139</td><td>&8B</td><td>|!|K</td><td>PLD</td><td>Ignored</td></tr>
<tr><td>140</td><td>&8C</td><td>|!|L</td><td>PLU</td><td>Ignored</td></tr>
<tr><td>141</td><td>&8D</td><td>|!|M</td><td>RI</td><td>Reverse index, move cursor up</td></tr>
<tr><td>142</td><td>&8E</td><td>|!|N</td><td>SS2</td><td>Map G2 set to GL (single shift)</td></tr>
<tr><td>143</td><td>&8F</td><td>|!|O</td><td>SS3</td><td>Map G3 set to GL (single shift)</td></tr>
<tr><td>144</td><td>&90</td><td>|!|P</td><td>DCS</td><td>Device Control String introducer</td></tr>
<tr><td>145</td><td>&91</td><td>|!|Q</td><td>PU1</td><td>Ignored</td></tr>
<tr><td>146</td><td>&92</td><td>|!|R</td><td>PU2</td><td>Ignored</td></tr>
<tr><td>147</td><td>&93</td><td>|!|S</td><td>STS</td><td>Ignored</td></tr>
<tr><td>148</td><td>&94</td><td>|!|T</td><td>CCH</td><td>Ignored</td></tr>
<tr><td>149</td><td>&95</td><td>|!|U</td><td>MW</td><td>Ignored</td></tr>
<tr><td>150</td><td>&96</td><td>|!|V</td><td>SPA</td><td>Ignored</td></tr>
<tr><td>151</td><td>&97</td><td>|!|W</td><td>EPA</td><td>Ignored</td></tr>
<tr><td>152</td><td>&98</td><td>|!|X</td><td></td><td>Ignored</td></tr>
<tr><td>153</td><td>&99</td><td>|!|Y</td><td></td><td>Ignored</td></tr>
<tr><td>154</td><td>&9A</td><td>|!|Z</td><td></td><td>Ignored</td></tr>
<tr><td>155</td><td>&9B</td><td>|!|[</td><td>CSI</td><td>Command Sequence Introducer</td></tr>
<tr><td>156</td><td>&9C</td><td>|!|\</td><td>ST</td><td>String Terminator</td></tr>
<tr><td>157</td><td>&9D</td><td>|!|]</td><td>OSC</td><td>As DCS</td></tr>
<tr><td>158</td><td>&9E</td><td>|!|^</td><td>PM</td><td>As DCS</td></tr>
<tr><td>159</td><td>&9F</td><td>|!|_</td><td>APC</td><td>As DCS</td></tr>
</table>

Appendix B

VT320/VT100 Escape sequences

Display attributes

CSI Ps;Ps...m	Select graphic rendition	
	Ps = 0	all attributes off
	Ps = 1	bold
	Ps = 2	faint
	Ps = 3	italics
	Ps = 4	underscore
	Ps = 5	blink
	Ps = 6	blink
	Ps = 7	reverse video
	Ps = 22	bold off
	Ps = 24	underscore off
	Ps = 25	blink off
	Ps = 26	blink off
	Ps = 27	reverse video off
	Ps = 30-37	foreground colour 30+colour
	Ps = 40-47	background colour 40+colour
	colour: 0=black, 1=red, 2=green, 3=yellow, 4=blue, 5=magenta, 6=cyan, 7=white	

Line attributes

ESC #3	Set double-width/double-height line, top half
ESC #4	Set double-width/double-height line, bottom half
ESC #5	Set single-width/single-height line
ESC #6	Set double-width/single-height line

Editing commands

CSI Pn A	Cursor up Pn lines, does not scroll
CSI Pn B	Cursor down Pn lines, does not scroll
CSI Pn C	Cursor right Pn columns, staying on same line
CSI Pn D	Cursor left Pn columns, staying on same line
CSI Pn a	ANSI cursor right Pn columns
CSI Pn e	ANSI cursor down Pn rows

CSI Pc G	ANSI cursor to absolute column Pc
CSI Pr d	ANSI cursor to absolute row Pr
CSI Pl;Pc f	Cursor to position Pl, Pc
CSI Pl;Pc H	Cursor to position Pl, Pc, as CSI Pl;Pc f
CSI Pn I	Cursor forward Pn tab stops
ESC D	Index. Cursor down one line, scrolling
ESC M	Reverse index. Cursor up one line, scrolling
CSI Pn F	Reverse index. Cursor up Pn lines, scrolling
ESC E	Cursor to start of next line, scrolling
ESC 7	Save cursor position and attributes
CSI s	Save cursor position and attributes, as ESC 7
ESC 8	Restore cursor position and attributes
CSI u	Restore cursor position and attributes, as ESC 8
CSI Ps J	Erase in display Ps = 0 cursor to end of screen, inclusive Ps = 1 start of screen to cursor, inclusive Ps = 2 entire screen, cursor does not move
CSI ? Ps J	Selective erase in display, Ps as for CSI Ps J
CSI Ps K	Erase in line Ps = 0 cursor to end of line, inclusive Ps = 1 start of line to cursor, inclusive Ps = 2 entire line, cursor does not move
CSI ? Ps K	Selective erase in line, Ps as for CSI Ps K

CSI Ps " q	Character protection attribute
	Ps = 0 erase protection off
	Ps = 1 character not erasable
	Ps = 2 character is erasable

CSI Pn X	Erase next Pn characters, inclusive
CSI Pn @	Insert Pn spaces at cursor
CSI Pn P	Delete Pn characters left of cursor, inclusive
CSI Pn L	Insert Pn lines before cursor line
CSI Pn M	Delete Pn lines below cursor line, inclusive
CSI ? Pn X	Erase next Pn characters, inclusive
CSI ? Pn @	Insert Pn spaces at cursor
CSI ? Pn P	Delete Pn characters left of cursor, inclusive
CSI ? Pn L	Insert Pn lines before cursor line
CSI ? Pn M	Delete Pn lines below cursor line, inclusive

Set/reset mode

`CSI Ps;Ps...h`	Set mode
`CSI Ps;Ps...l`	Reset mode

	Set mode (h)	Reset mode (l)
Ps = 2	keyboard locked	keyboard unlocked
Ps = 3	receive esc sequences	ignore esc sequences
Ps = 4	character insert mode	character replace mode
Ps = 10	horizontal editing	horizontal editing off
Ps = 12	local echo off	local echo on
Ps = 20	newline mode on	newline mode off

`CSI ?Ps;Ps...h`	Set DEC mode
`CSI ?Ps;Ps...l`	Reset DEC mode

	Set mode (h)	Reset mode (l)
Ps = 1	cursor app. mode	cursor key mode
Ps = 2	VT320/102 emulation	VT52 emulation
Ps = 3	132 columns	80 columns
Ps = 4	smooth scrolling	normal jump scrolling
Ps = 5	screen reverse video	screen normal video
Ps = 6	origin to scroll region	origin to whole screen
Ps = 7	auto-wrap on	auto-wrap off
Ps = 8	auto-repeat on	auto-repeat off
Ps = 18	print term. form feed	no print termination
Ps = 19	print extent full screen	print extent scroll region
Ps = 25	cursor on	cursor off
Ps = 34	macro TERMINALS	macro TERMINALR
Ps = 38	Tek graphics mode	Tek text mode
Ps = 42	enable NRCS	disable NRCS
Ps = 66	keypad app. mode	keypad mode
Ps = 68	data process	typewriter

Character sets

`ESC (c`	Load 94-byte character set to G0
`ESC )c`	Load 94-byte character set to G1
`ESC *c`	Load 94-byte character set to G2
`ESC +c`	Load 94-byte character set to G3
`ESC -c`	Load 96-byte character set to G1
`ESC .c`	Load 96-byte character set to G2
`ESC /c`	Load 96-byte character set to G3

c	size	character set
A	94	British (VT102 only)
A	96	ISO Latin-1 (default in G2/G3)
B	94	ASCII (default in G0 and G1)
C	94	Finnish
E	94	Norwegian/Danish
H	94	Swedish
K	94	German
Q	94	French Canadian
R	94	French
Y	94	Italian
Z	94	Spanish
4	94	Dutch
5	94	Finnish
6	94	Norwegian/Danish
7	94	Swedish
9	94	French Canadian
%6	94	Portuguese
=	94	Swiss
`	94	Norwegian/Danish
0	94	DEC special graphics
1	94/96	ALT-ROM
2	94	Alternative graphics
%5	94	DEC supplemental graphics
<	94	User preferred supp. set UPSS
>	94	DEC technical set

ESC n	Lock shift character set in G2 to GL
ESC o	Lock shift character set in G3 to GL
ESC ~	Lock shift character set in G1 to GR
ESC }	Lock shift character set in G2 to GR
ESC |	Lock shift character set in G3 to GR
ESC N	Single shift character set in G2 to GL
ESC O	Single shift character set in G3 to GL

Miscellaneous

`ESC # 8`	Test screen alignment (fill screen with E's)	
`CSI 2;Ps v`	Invoke test `Ps`	
`CSI Pt;Pb r`	Set top and bottom scrolling margins, `Pt` & `Pb`	
`ESC =`	Set application keypad mode	
`ESC >`	Set numeric keypad mode	
`ESC H`	Set horizontal tab at cursor position	
`CSI Ps g`	Clear tabs	
	`Ps` = 0	clear tab at cursor position
	`Ps` = 3	clear all tabs
`ESC c`	Hard reset of terminal	
`CSI ! p`	Soft reset of terminal	
`CSI Ps;Ps...q`	Control LEDs	
	`Ps` = 0	clear all LEDs
	`Ps` = 1,2,3,4	set LED 1, 2, 3 or 4
`CSI Pl;Pc " p`	Set terminal type	
	`Pl` = 61, `Pc` = 0	VT102, 7-bit controls
	`Pl` = 62, `Pc` = 0	VT220, 8-bit controls
	`Pl` = 62, `Pc` = 1	VT220, 7-bit controls
	`Pl` = 62, `Pc` = 2	VT220, 8-bit controls
	`Pl` = 63, `Pc` = 0	VT320, 8-bit controls
	`Pl` = 63, `Pc` = 1	VT320, 7-bit controls
	`Pl` = 63, `Pc` = 2	VT320, 8-bit controls
`ESC sp F`	Disable output of 8-bit controls	
`ESC sp G`	Enable output of 8-bit controls	
`CSI Pn i`	Media copy	
	`Pn` = 0	print entire screen
	`Pn` = 4	exit printer controller mode
	`Pn` = 5	enter printer controller mode
`CSI ? Pn i`	DEC media copy	
	`Pn` = 1	print cursor line
	`Pn` = 4	exit auto print mode
	`Pn` = 5	enter auto print mode

Reports and requests

`CSI c`	Request primary device attributes. Response:	
	`CSI ? 1 c`	VT100 terminal
	`CSI ? 6 c`	VT102 terminal
	`CSI ? 62 c`	VT220 terminal
	`CSI ? 63;1;2;8;9 c`	VT320 terminal

ESC Z	Request primary device attributes. As CSI c
CSI > c	Request secondary device attributes. Response: CSI > 24;0;0;0 c VT320, version 0.0
CSI 5 n	Request operating status. Response: CSI 0 n OK
CSI 6 n	Request cursor position. Response: CSI Pr;Pc R where Pr is row, Pc is col
CSI ? 15 n	Request printer status. Response: CSI ? 10 n printer ready CSI ? 11 n printer not ready CSI ? 13 n printer not connected
CSI ? 25 n	User definable key status. Response: CSI ? 20 n keys are unlocked CSI ? 21 n keys are locked

CSI ? 26 n	Request keyboard dialect. Response:
	CSI ? 27;Ps n
	Ps = 1 ASCII
	Ps = 2 British
	Ps = 4 French Canadian
	Ps = 6 Finnish
	Ps = 7 German
	Ps = 8 Dutch
	Ps = 9 Italian
	Ps = 11 Swiss
	Ps = 12 Swedish
	Ps = 13 Norwegian/Danish
	Ps = 14 French
	Ps = 15 Spanish
	Ps = 16 Portuguese

CSI 1 $ u	Request terminal state. Response: DCS 1 $ ST
DCS Ps $ p s ST	Terminal restore state. No response

CSI & u	Request UPSS. Response:
	DCS Ps ! u string ST
	Ps = 0 for 94-byte set, or 1 for 96-byte set
	s = A for ISO Latin, or %5 for DEC Supp.

DCS Ps ! u s ST	Assign User Preferred Supplemental Set. Ps and s are the same as CSI & u request
CSI 1 $ w	Request cursor information. Response: DCS 1 $ u Pr;Pc;Pp;Srend;Satt;Sflag;Pgl;Pgr;Scss;Sdesig ST Pr = cursor row, Pc = cursor column, Pp = 1

	Srend = 40h	+ 8 (rev video on) + 4 (blink on)
		+ 2 (underline on) + 1 (bold on)
	Satt = 40h	??? (selective erase)
	Sflag = 40h	+ 8 (autowrap) + 4 (SS3)
		+ 2 (SS2) + 1 (origin mode on)
	Pgl = char set in GL (0 = G0, 1 = G1, 2 = G2, 3 = G3)	
	Pgr = char set in GR (0 = G0, 1 = G1, 2 = G2, 3 = G3)	
	Scss = 40h	+ 8 (G3 is 96 char), + 4 (G2 is 96 char)
		+ 2 (g1 is 96 char), + 1 (G0 is 96 char)
	Sdesig = string of character idents for sets G0 to G3	
CSI 2 $ w	Request tab stop information. Response:	
	DCS 2 $ u Pc;Pc... ST	
	Pc = column number where tab stop occurs	
DSC Ps $ t s ST	Restore presentation state. No response.	
	Ps = 1	for cursor info, s as in DSC 1 $ w
	Ps = 2	for tab info, s as in DSC 2 $ w
CSI Pa $ p	Request state of ANSI mode controls. Response:	
	CSI Pa;Ps $ y	
	Pa = 2	keyboard action (if locked)
	Pa = 3	control representation (no debug)
	Pa = 4	insert/replace mode (if insert mode)
	Pa = 10	horizontal editing (perm reset)
	Pa = 12	send/receive (local echo on)
	Pa = 20	newline (in newline on)
	Ps = 0	unknown mode
	Ps = 1	set
	Ps = 2	reset
	Ps = 3	permanently set
	Ps = 4	permanently reset
CSI ? Pd $ p	Request state of DEC modes. Response:	
	CSI Pd;Ps $ y	
	Pd = 1	cursor key mode
	Pd = 2	ANSI mode
	Pd = 3	132 columns
	Pd = 4	smooth scrolling
	Pd = 5	reverse video
	Pd = 6	origin mode
	Pd = 7	auto wrap
	Pd = 8	auto repeat
	Pd = 18	print with form feed
	Pd = 19	print extent full screen

Pd = 25	text cursor enabled
Pd = 42	DEC NRCS in use
Pd = 66	numeric keypad
Pd = 67	destructive backspace
Pd = 68	keyboard usage

Ps as above

`DCS $ q s ST` Request control function setting.

s = $}	select active status display
s = “q	set character attribute
s = “p	set conformance level
s = $~	set status line type
s = r	set top and bottom margins
s = m	set graphic rendition

Response:

`DCS Ps $ r s ST`

Ps = 0	valid request
Ps = 1	invalid request

The response is the same as the command except that the leading CSI is omitted.

`CSI sol x` Request terminal parameters (VT102 only)

sol = 0	terminal can send unsolicited reports
sol = 1	terminal reports only on request
sol = 2	this is a report
sol = 3	terminal reporting only on request

Response:

`CSI sol;par;nbits;xspeed;rspeed;clkmul;flags x`

par = 1	no parity
par = 2	space parity
par = 3	mark parity
par = 4	odd parity
par = 5	even parity
nbits = 1	8 bits/character
nbits = 2	7 bits/character
xspeed =	transmit speed index
rspeed =	receive speed index

0 = 50, 8 = 75, 16 = 110, 24 = 134.5, 32 = 150, 40 = 200, 48 = 300, 56 = 600, 64 = 1200, 72 = 1800, 80 = 2000, 88 = 2400, 96 = 3600, 104 = 4800, 112 = 9600, 120 = 19200, 128 = 38400

clkmul = 1 clock rate multiplier is 16

flags = 0

Download function keys

```
DSC Pc;Pl|Kyl/Stl;... ST
```

Pc = 0	clear all keys
Pc = 1	do not clear keys
Pl = 0	lock keys
Pl = 1	do not lock keys
Kyl = key number (decimal)	
Stl = string (hex)	

For example, the sequence `DSC 1;1|28/48656c70 ST` defines the DEC Help key (number 28) to generate the word `Help`.

Download character set

```
DCS Pfn;Pcn;Pe;Pcms;Pw;Pt { Dscs Sxbpl;Sxbp2;...;Sxbpn ST
```

Pfn	font number (0 or 1)
Pcn	start character number
Pe = 0	erase all characters in this set
Pe = 1	erase only characters being loaded
Pe = 2	erase all characters in all sets
Pcms = 0	default matrix size (7 x 10)
Pcms = 2	5 x 10 matrix
Pcms = 3	6 x 10 matrix
Pcms = 4	7 x 10 matrix
Pw = 0	default width (80 columns)
Pw = 1	80 column width
Pw = 2	132 column width
Pt = 0	device default (text)
Pt = 1	text
Pt = 2	full-cell
Dscs	character set name
Sxbp	sixel bit patters for characters

VT320/VT102 Sequences transmitted

In the VT320 and VT102 terminal emulators the keypad keys and cursor keys are defined to transmit standard escape sequences to the host. In addition, in the VT320 terminal the function keys also transmit special sequences. The use made of these sequences depends entirely on the software running on the host.

VT320 Numeric keypad keys

VT320 key	Archimedes key	Numeric keypad	Application keypad
PF1	Num Lock	SS3 P	SS3 P
PF2	/	SS3 Q	SS3 Q
PF3	*	SS3 R	SS3 R
PF4	#	SS3 S	SS3 S
keypad 7	keypad 7	7	SS3 w
keypad 8	keypad 8	8	SS3 x
keypad 9	keypad 9	9	SS3 y
keypad 4	keypad 4	4	SS3 t
keypad 5	keypad 5	5	SS3 u
keypad 6	keypad 6	6	SS3 v
keypad 1	keypad 1	1	SS3 q
keypad 2	keypad 2	2	SS3 r
keypad 3	keypad 3	3	SS3 s
keypad 0	keypad 0	0	SS3 p
keypad .	keypad .	.	SS3 n
keypad Enter	keypad Enter	|M	SS3 M
keypad ,	keypad +	+	SS3 l
keypad -	keypad -	-	SS3 m

VT320 Cursor keys

VT320 key	Archimedes key	Cursor key mode	Application mode
cursor up	cursor up	CSI A	SS3 A
cursor down	cursor down	CSI B	SS3 B
cursor right	cursor right	CSI C	SS3 C
cursor left	cursor left	CSI D	SS3 D

VT320 editing keys

VT320 key	Archimedes key	Sequence transmitted
Find	Action Home	CSI 1~
Insert Here	Action Insert	CSI 2~
Remove	Action Delete	CSI 3~
Select	Action Copy	CSI 4~
Prev Screen	Action Page Up	CSI 5~
Next Screen	Action Page Down	CSI 6~

VT320 function keys

VT320 key	Archimedes key	Sequence transmitted
f6	Action F1	CSI 17~
f7	Action F2	CSI 18~
f8	Action F3	CSI 19~
f9	Action F4	CSI 20~
f10	Action F5	CSI 21~
f11	Action F6	CSI 23~
f12	Action F7	CSI 24~
f13	Action F8	CSI 25~
f14	Action F9	CSI 26~
Help	Action F10	CSI 28~
Do	Action F11	CSI 29~
f17	Action F12	CSI 31~
f18	Action Print	CSI 32~
f19	Action Scroll Lock	CSI 33~
f20	Action Break*	CSI 34~

Appendix C

VT52 Escape sequences

`ESC <`	Enter ANSI emulation
`ESC =`	Set application keypad mode
`ESC >`	Set numeric keypad mode
`ESC A`	Cursor up
`ESC B`	Cursor down
`ESC C`	Cursor right
`ESC D`	Cursor left
`ESC H`	Cursor to home position
`ESC I`	Reverse line feed
`ESC Y Pl;Pc`	Cursor to line `Pl`, column `Pc`
`ESC J`	Erase to end of screen
`ESC K`	Erase to end of line
`ESC F`	Select special graphics character set
`ESC G`	Select US (ASCII) character set
`ESC V`	Print cursor line
`ESC W`	Enter printer controller mode
`ESC X`	Exit printer controller mode
`ESC Z`	Identify terminal. Response: `ESC / Z`
`ESC ]`	Print screen
`ESC ^`	Enter auto print mode
`ESC _`	Exit auto print mode
`ESC 7`	Save cursor position
`ESC 8`	Restore cursor position

VT52 Sequences transmitted

VT52 Numeric keypad keys

VT52 key	Archimedes key	Numeric keypad	Application keypad
PF1	Num Lock	ESC P	ESC ? P
PF2	/	ESC Q	ESC ? Q
PF3	*	ESC R	ESC ? R
PF4	#	ESC S	ESC ? S
keypad 7	keypad 7	7	ESC ? w
keypad 8	keypad 8	8	ESC ? x
keypad 9	keypad 9	9	ESC ? y
keypad 4	keypad 4	4	ESC ? t
keypad 5	keypad 5	5	ESC ? u
keypad 6	keypad 6	6	ESC ? v
keypad 1	keypad 1	1	ESC ? q
keypad 2	keypad 2	2	ESC ? r
keypad 3	keypad 3	3	ESC ? s
keypad 0	keypad 0	0	ESC ? p
keypad .	keypad .	.	ESC ? n
keypad Enter	keypad Enter	|M	ESC ? M
keypad ,	keypad +	+	ESC ? l
keypad -	keypad -	-	ESC ? m

VT52 Cursor keys

VT52 key	Archimedes key	Cursor key mode	Application mode
cursor up	cursor up	CSI A	SS3 A
cursor down	cursor down	CSI B	SS3 B
cursor right	cursor right	CSI C	SS3 C
cursor left	cursor left	CSI D	SS3 D

Appendix D

Archives

In many cases you will find that a file you have downloaded has been archived. An archive is a single or group of files or directories that have been compressed into a single file. An archived file takes less time to download because it is smaller, and it is easier to handle than a group of separate files.

Normally you will be told before downloading, whether a file has been archived. Once you have received the file, you must de-archive it using the application SparkPlug which is supplied on the Hearsay Extras disc. To de-archive a file, load SparkPlug onto the icon bar, and simply double-click on the archive file. You can do this from the RX batch window. Sparkplug will now display the files in the archive in a window from where you can drag them to the required destination. Once all the files have been extracted, the archive may be removed from the batch.

If the downloaded archive does not have the correct filetype, you must drag it to the SparkPlug icon on the icon bar to un-archive it. Alternatively, you may set it's filetype to &DDC.

Creating archives

SparkPlug may only be used to extract files from archives, so if you wish to create archives you will need the full archiving program Spark. Details of where to get this program from are given in the SparkPlug **Info** box. Alternatively you can create archives using the non-WIMP archive program Arc which is available on the Hearsay Extras disc in directory `Arc`. Full details of how to use this program are given in the file `Arc.ReadMe`.

Appendix E

Function key definitions

All terminals

	Function key	Shift function key
F1	Quick setup	Send answerback string
F2	Telephone directory	Re-dial last number
F3	Save capture buffer	Save sprite
F4	Connect	Disconnect
F5	Receive file	Send file
F6	Display RX batch	Display TX batch
F7	Edit macros	Toggle status line
F8	Send short break	Send long break

Text terminals only

	Function key	Shift function key
F9	Spooling on	Spooling off
F10	Hold line (Ctrl S)	Release line (Ctrl Q)
F11	Auto-print on	Auto-print off
F12	* commands	

Viewdata terminals only

	Function key	Shift function key
F9	Tag frame	Show tagged frames
F10	Go to tagged frame	Paste tagged frame
F11	Step down tagged frame	Step up tagged frame
F12	* commands	

Viewdata editor

	Function key	Shift function key
F1	Alphanumeric red	Graphics red
F2	Alphanumeric green	Graphics green
F3	Alphanumeric yellow	Graphics yellow
F4	Alphanumeric blue	Graphics blue
F5	Alphanumeric magenta	Graphics magenta
F6	Alphanumeric cyan	Graphics cyan
F7	Alphanumeric white	Graphics white
F8	FlashSteady	
F9	Double height	Single height
F10	Separated graphics	Contiguous graphics
F11	New background	Black background
F12	* commands	

Appendix F

Modem drivers

Modem driver functions

Hearsay modem drivers are script files located in directory !Hearsay.Driver. Modem drivers may call the functions listed below.

```
int internal_link(int link);
```

internal_link sets the type of link level protocol used internally by Hearsay. The parameter link should be set to 0 for none, 1 for Vasscom, or 2 for MNP. It returns the type in use.

```
int online(void);
```

online returns 1 indicating Hearsay is online, or 0 for offline.

```
void onlinechange(int online);
```

onlinechange is used to signal to Hearsay that a modem has gone online or offline, and allows it to update it's values. The parameter online should be set to 1 for online, or 0 for offline. Normally, this function will only be used used to signify that the modem has gone online. Going offline is handled by Hearsay.

```
int devcon(fn, port, pl, p2);
```

devcon is used to call low-level serial port functions. The parameter fn specifies the function to be called, and may be one of the following:

Device_TxRate
Device_RxRate
Device_ParityBits
Device_DataBits
Device_StopBits
Device_Status
Device_Claim
Device_Flow
Device_Select
Device_Break
Device_Get
Device_Put
Device_CountPurge
Device_Channel

The parameters p1 and p2 are passed to the function.

Please note that this function is not built into Hearsay, but is defined in the library files !Serial and SerialDev. !Serial redirects the function call to the internal serial port and !SerialDev redirects the call to the current block driver. See *Appendix H* for more details.

Functions provided by the modem drivers

The following is a skeleton program showing all the functions that should be provided in a modem driver. Please refer to one of the drivers provided for more details.

```
// count or purge buffer
int modem_countpurge(int rxtx,int code)
{
  return(devcon(Device_CountPurge,modemport,rxtx,code));
}
// generate break for time cs
void modem_break(int time)
{
  devcon(Device_Break,modemport,time,0);
}
// tell modem to use a given sort of flow control
void modem_setflow(int flow)
{
  devcon(Device_Flow,modemport,flow,1);
  modemflow=flow;
}
// read flow control in use
int modem_readflow(void)
{
  modemflow=devcon(Device_Flow,modemport,0,0);
  return(modemflow);
}
// tell modem to use a given link level protocol
int modem_link(int link)
{
  modemlink=internal_link(link);
  return(modemlink);
}
// read rx baud rate at which modem is talking to
outside world
int modem_readrxrate(void)
{
  return(modemrxrate);
}
```

```
// set rx rate
void modem_setrxrate(int rate)
{
  devcon(Device_RxRate,modemport,rate,1);
  modemrxrate=devcon(Device_RxRate,modemport,0,0);
}
// read tx baud rate at which modem is talking to
outside world
int modem_readtxrate(void)
{
  return(modemtxrate);
}
// set tx rate
void modem_settxrate(int rate)
{
  devcon(Device_TxRate,modemport,rate,1);
  modemtxrate=devcon(Device_TxRate,modemport,0,0);
}
```

Often, communication between the modem and the computer will take place at a fixed baud rate, for example 9600. Whilst the modem may use a different rate for communicating with the outside world. The four functions described above allow the modem to set the hardware to one rate whilst reporting a different baud rate back to Hearsay. In general it is important that the true baud rate is reported to Hearsay, since certain operations require to know this. For example the FTP's calculate how long transfers will take from the baud rate i.e. the rate bytes actually get sent at, not how fast they reach the modem.

```
// read number of data bits
int modem_readbits(void)
{
  return(modembits);
}
// set number of data bits
void modem_setbits(int bits)
{
  devcon(Device_DataBits,modemport,bits,1);
  modembits=devcon(Device_DataBits,modemport,0,0);
  modemparity=devcon(Device_ParityBits,modemport,0,0);
  modemstop=devcon(Device_StopBits,modemport,0,0);
}
```

```
// read parity bits
int modem_readparity(void)
{
  return(modemparity);
}
// set parity bits
void modem_setparity(int parity)
{
  devcon(Device_ParityBits,modemport,parity,1);

  modembits=devcon(Device_DataBits,modemport,0,0);
  modemparity=devcon(Device_ParityBits,modemport,0,0);
  modemstop=devcon(Device_StopBits,modemport,0,0);
}
// read stop bits
int modem_readstop(void)
{
  return(modemstop);
}
// set stop bits
void modem_setstop(int stop)
{
  devcon(Device_StopBits,modemport,stop,1);

  modembits=devcon(Device_DataBits,modemport,0,0);
  modemparity=devcon(Device_ParityBits,modemport,0,0);
  modemstop=devcon(Device_StopBits,modemport,0,0);
}
// read answer/originate mode
int modem_readanswer(void)
{
  return(modemanswer);
}
// set answer/originate mode
void modem_setanswer(int answer)
{
  modemanswer=answer;
}
```

```
// set to tone/pulse dial
int modem_tonedial(int tonedial)
{
  modemtonedial=tonedial;
  return(modemtonedial);
}

// is the modem on line ?
int modem_online(void)
{
  return(devcon(Device_Status,modemport,0,-1) &
SERIAL_DCD);
}

void modem_autoanswer(int rings)
{
...
}

// dial a number and attempt to go on line
void modem_dial(string number)
{
...
}

// reconnect
void modem_reconnect(void)
{
...
}

// talk to modem
void modem_talk(void)
{
...
}

// put modem off line
void modem_disconnect(void)
{
...
}

// put modem on line
void modem_connect(void)
{
...
}
```

```
// called when Hearsay is about to terminate
// good time to hang up line etc.
void modem_terminate(void)
{
  devcon(Device_Select,modemport,0,0);
}
// cancel modem operation
void modem_interrupt(void)
{
...
}
// called when driver is installed
// will be followed by call to set baud rates/bits
// return 0 if successful
int modem_initiate(int port)
{
  modemport=port;

  if(!devcon(Device_Claim,modemport,0,0)) return(2);

  devcon(Device_Select,modemport,1,0);

  devcon(Device_Channel,modemport,0,0);

  devcon(Device_RxRate,modemport,2400,1);
  devcon(Device_TxRate,modemport,2400,1);

  modem_setbits(8);
  modem_setparity(0);
  modem_setstop(1);

  setpoll(0);

  sprints("ATQ0V1S0=0|M");
  pause(50);

  setpoll(1);

  return(0);
}
```

Appendix G

Acorn Device Claim Protocol

Hearsay supports the Acorn device claim protocol which allows applications to amicably share devices like the serial port. So two applications which both support this protocol may be installed on the icon bar at the same time.

However, only one program will be active and have control over the serial port and the modem; the other program is in a quiescent state i.e. a non-active state. In this non-active state, the icon bar icon will be shaded.

Programs can be toggled in and out of the active state by clicking on the icon bar icon with Adjust.

If something attempts to claim the active application's device when it is online or when a file transfer is going on, it will veto the request.

When Hearsay is quit when online or when a file transfer is active, a dialogue box will appear requesting confirmation. A potential problem with this, is that people using the null modem driver (or with incorrect modem wiring) are always online, and will always be prompted when quitting Hearsay. A possible solution to this is to disable confirmation prompts using the **Confirm box** option from the **Choices** dialogue box.

Appendix H

Device block drivers

Hearsay may be used directly with the internal serial port or with the application `!SerialDev` and the device block drivers.

Internal serial port

Unless you have an expansion serial port, you will want to use the internal serial port. Hearsay is supplied by default to use this port and no changes are necessary.

Block drivers

If you have an expansion serial port you may use the block drivers to use this port instead of the internal one. To do this you must have a copy of `!SerialDev` containing a suitable block driver for your device; this should be supplied with the expansion board. To use the block drivers, make the following changes to Hearsay:

1. Move the file `!SerialDev` from the `!Hearsay` directory into the `Library` directory.
2. Move the file `!Serial` out of the `Library` directory into the `!Hearsay` directory.

   Steps 1 and 2 above swap the positions of the files `!SerialDev` and `!Serial`.
3. Edit `!SerialDev` so that the function:

   ```
   setserialdev(name, port);
   ```

   found towards the bottom of the file, contains the name of the appropriate driver and sets the required port.

Please make sure that the application `!SerialDev` (which contains the block drivers) is 'seen' by the desktop before Hearsay is run.

It is possible to use the block drivers to drive the internal serial port, but this is **not recommended** because some versions of the block driver may not work with block read/write operations. Furthermore the device drivers do not provide the extended buffering that the normal Hearsay internal driver provides.

Appendix I

Glossary

ANSI

The American National Standards Institution. The ANSI terminal emulator conforms to the ANSI terminal emulation standard.

Answer Mode

The mode of operation which sets the frequency for data communication. If one end of a communications link is set to Answer Mode, the other end should be set to Originate Mode. In 1200/75 operation, Answer Mode sets the transmit baud rate to 1200, and the receive baud rate to 75.

Application Mode

The terminal emulator mode which causes the numeric keypad to send special escape sequences.

ASCII

The American Standard Code for Information Interchange. This standard defines the character set used by most computers and communications systems. The ASCII standard represents control codes, printable characters etc., as 7-bit characters ranging from 0 to 127.

Auto answer

The feature of a modem that allows it to automatically answer a call and set up a communications link (i.e. Connect the call). This type of modem is essential for use in unattended host systems.

Auto dial

The feature of a modem that makes it capable of making telephone calls without the use of a telephone.

Auto repeat

The feature whereby a key held down for a short time repeats the character at regular intervals, until the key is released.

Auto wrap

The feature of a terminal emulator which causes a carriage return and line-feed to be output when the cursor reaches the right margin.

Batch

The term used to describe a group of files that are to be transmitted over a communications link. The Ymodem file transfer protocol is capable of transferring a batch of files.

Baud Rate

The speed at which data is transmitted over a communications link. The speed is measured in bits/second, so a speed of 300 baud will send 30 characters/second (assuming word length is 10 bits).

Bulletin Board

A computer service that allows callers to access information, leave messages, download files etc. The bulletin board software that manages the service is normally left unattended to answer the phone (via an auto-answer modem) and control access to the facilities provided.

Capture buffer

A buffer used in scrolling text terminals to store lines of incomming data, and allowing the user to scroll back through that data.

Carrier

The tone transmitted by the modem on which the data is modulated.

C.E.T File Interchange Format

The Council for Educational Technology format for telesoftware downloading. This format is used by many Viewdata services.

Checksum

A byte sent at the end of a packet of data that is used to ensure that all the bytes in the packet are correct. If the checksum for the data received is incorrect, the data has been corrupted and the packet must be resent.

Control Code

A non-printable character that can have a special purpose in terminal emulations. ASCII control codes have values between 0 and 31.

CTS

Clear To Send. A serial signal received by a terminal indicating that it may send data.

Data Format

The collective name for word length, number of stop bits and parity of the data transmitted.

Data Communications Equipment (DCE)

The term used by the CCITT (International Consultative Committee for Telegraph and Telephones) to describe communications equipment i.e. modems.

Data Terminal Equipment (DTE)

The term used by the CCITT (International Consultative Committee for Telegraph and Telephones) to describe computer terminals.

DCD

Data Carrier Detect. A serial signal sent by modem to indicate that a carrier is being received.

Download

The term used to describe the transfer of data or programs from a remote computer to the user.

DSR

Data Set Ready. A serial signal from a DCE indicating that it is ready to communicate.

DTR

Data Terminal Ready. A serial signal from a DTE indicating that it is ready to communicate.

Duplex

A transmission system where data is transmitted in both directions simultaneously. Also called Full Duplex.

Electronic Mail

The use a communications link to transfer messages between users.

Escape Sequence

A sequence of characters used to transmit special instructions to a terminal to control certain aspects of its operation. An escape sequence starts with the ESC character (ASCII character 27), and is followed by a further character or sequence of characters.

File transfer protocol

A standard method of transmitting a file using error checking to ensure thereare no error in the transferred file.

Filter

The feature of communications software that traps and removes unwanted characters such as control codes.

Flow Control

The method used by communications software to limit the rate at which data is received, to limit that it can be processed. One standard method of implementing flow control is to use XON and XOFF characters.

Frame

The standard page of information used in Viewdata terminals. A frame consists of 25 lines of 40 characters per line.

Handshake

The exchange of signals (e.g. CTS/RTS), to establish the readiness of the DCE and DTE to send and receive data.

Half Duplex

A transmission system where data is transmitted in both directions, but only in one direction at a time.

Hard Copy

The term used to describe printed copy of text or graphics output. Hard copy is usually produced on dot-matrix, daisywheel or laser printers.

Hayes

The trademark of a modem manufacturer and the most widely used protocol for controlling intelligent modems.

Host system

The computer system to which a terminal is connected. Host systems are usually powerful mainframe computers that can communicate with a number of terminals at the same time.

Kermit

A file transfer protocol developed at Columbia University USA for transferring files between microcomputers and mainframe machines. It is a batch transfer protocol sending 8 bit files over 7 lines, and preserves file lengths and sends filenames along with the file.

Link level error correction

A method of transferring error free data used when communicating at high speeds. It works by sending blocks of data with a checksum at the end of each block, and retransmitting the block is there is an error.

Local Echo

The feature of a terminal to reflect the characters sent to the terminal screen. Normally local echo is not necessary as the host computer echoes characters back.

Modem

A device used to transfer data across the telephone network. The device is called a Modem because it MOdulates and DEModulates the data. Modulation is the process of converting digital data from the computer to an analogue signal suitable for transmission over the telephone network. Demodulation is the reverse of this, and converts incoming analogue signals into digital data suitable for use by a computer.

MNP

A link level error correction protocol.

Newline Mode

The mode of operation where a transmitted Carriage Return character (CR) is interpreted as a Carriage Return followed by a Line Feed (CR/LF). Also, a received Line Feed character (LF) is interpreted as a Carriage Return followed by a Line Feed (CR/LF).

Offline

The state of a terminal not connected to any host system.

Online

The state of a computer terminal when it is connected to a host system, and a data link established.

Originate Mode

The mode of operation which sets the frequency for data communication. If one end of a communications link is set to Originate Mode, the other end should be set to Answer Mode. In 1200/75 operation Originate Mode sets the transmit baud rate to 75, and the receive baud rate to 1200. Callers to Prestel and other 1200/75 host systems, should use this mode.

PABX

Private Automatic Branch Exchange. The term describing the equipment which manages internal telephone systems.

Packet

The term used to describe a block of data being transmitted over a data link. File transfer protocols such as Xmodem and Kermit transfer data in packets, so that if an error is detected during file transfer, the system need only re-transmit the corrupted packet. Packet sizes may vary in size, but normally they are 128 or 1024 bytes in length.

Parity Bit

An extra bit added to an ASCII character to provide a simple form of error checking. Parity is usually said to be EVEN or ODD. With EVEN parity, if the number of bits 'set' in a byte is odd, the eighth bit is modified to make it even. With ODD parity the same process is carried out to maintain an odd number of bits 'set' in the byte. It is essential that both ends of the data link are configured for the same type of parity.

Pixel

A dot on a computer screen which can be switched on or off, and on some systems can have a number of different intensities. The Tektronix terminal in Hearsay can operate at pixel level to produce graphics displays.

Prestel

A large computerised information service operated by British Telecom accessible via the public telephone network. Prestel stores many thousands of frames of information on a number of large computers situated at various locations throughout the UK. Once registered on Prestel you can access frames of information using a Viewdata terminal, and take advantage of other facilities such as telesoftware downloading, electronic mailbox, telex etc.

Protocol

A set of rules which both local and remote equipment must obey during communication. Protocols are particularly important for the transfer of files to ensure that both ends of the data link know exactly how the file is to be transferred i.e. the packet size, the type of error checking etc.

PSTN

Public Switched Telephone Network. The term describing the system used to make normal telephone calls.

Remote Computer

See Host System.

Response Frame

A blank space provided by a Viewdata service for the user to fill in particular details. Response frames are used by the remote computer to request details from the caller such as his/her name and address e.g. when ordering goods on a teleshopping service.

RS423

The standard which defines the electrical and physical characteristics of an asynchronous serial interface. The RS423 standard is compatible with the older RS232 standard.

RTS

Request To Send. A serial signal sent by a terminal indicating it is ready to send data.

SEAlink

A 'windowed' version of the XMODEM file transfer protocol.

Serial port

A type of electrical connection between computers and other devices. On the majority of computers the serial interface is used to connect the computer to a modem. The term used to describe the standard that defines serial connections is RS423 or RS232.

Spool

The process of transferring all information sent to the screen to disc or another filing system. Spooling is often used to keep a media copy of all data received from a remote computer.

Start Bit

An extra bit added to a character before it is transmitted to identify the start of the character. The RS423 interface automatically generates the start bit at the transmitting end of the data link, and strips it off at the receiving end.

Stop Bit

An extra 1 or 2 bits added to the end of each character transmitted to identify the end of the character. The RS423 interface automatically generates the stop bits at the transmitting end of the data link, and strips them off at the receiving end. It is essential that the serial interfaces at both ends are set to the same number of stop bits.

Tag buffer

A buffer where Viewdata frames may be temporarily stored for recall at a later date.

Tektronix

A high resoulution graphics terminal.

Telecom Gold

A large computerised information service operated by British Telecom available via the public telephone network. Telecom Gold is a scrolling text service allowing access to a whole range of databases worldwide, and includes Telex, electronic mail and software downloading facilities.

Telesoftware

Computer programs or files stored on the remote computer that are available for downloading by the caller. There is often a charge for the software, but many bulletin boards contain a wide range of free programs.

Teletype

A simple terminal emulator which recognises only a small set of control codes and no escape sequences (unlike VT100 for example). It should be used with hosts that do not specifically cater for any terminal emulations provided by the users terminal.

Terminal

The equipment used to communicate with the host computer. The term originally described a teletypewriter, keyboard and paper tape punch, but now usually means a VDU plus keyboard and disc drive.

Upload

The term used to describe the transfer of data or programs from the caller to the host computer.

Videotex

Another name for Viewdata

Viewdata

A terminal emulation utilising the teletext character set, which offers coloured text and graphics, and special features such as double-height characters, separated graphics and flash. Unlike scrolling text terminals such as VT100, Viewdata information is sent as frames of information; each frame being 24 lines of 40 characters. A Viewdata terminal must be used to access Viewdata services such as Prestel.

VT52

A popular terminal emulation, but the most primitive of the VT terminals in common use. The VT52 terminal has been mostly superseded by VT100.

VT102

A sophisticated terminal emulation which implements a large number of escape sequences and control codes. VT102 is one of the most widely used emulations, and is used, for example, to access the Telecom Gold service. VT102 is a development of VT100 offering a number of extra commands, and implementing most of the relevant ANSI codes.

VT320

A more modern and extended version of VT220.Word Length

The total number of bits which make up a unit of data transfer. The word length is not necessarily the same length as the word length within the computer, since it includes stop bits, start bits and parity bits.

Xon/Xoff

A simple method of providing flow control i.e. limiting the rate at which data is received, to the the rate at which it can be handled. When the receiving computer wants to stop the transmitting computer from sending further data, it sends an XOFF (CTRL S) character. To resume it sends an XON character (CTRL Q).

Xmodem

A sophisticated file transfer protocol that allows transfer of any type of file. It is an 8 bit transfer, and usually sends data in 128 byte packets. The normal error checking is by checksums, although most XMODEM transfers now allow CRC (Cyclic Redundancy Check) checking as well.

Ymodem

A file transfer protocol developed from XMODEM to allow variable length packets. YMODEM uses 1024 byte packets, and provides both CRC and checksum error checking. YMODEM is not compatible with XMODEM.

Zmodem

A modern file transfer protocol.

Index to CScript functions

Index

Scanned by Rick
2015/01/01

http://www.heyrick.co.uk/blog/