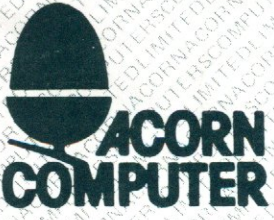
The Econet System User Guide



THE ECONET SYSTEM USER GUIDE

17 March 1983
403,100 Issue 2

The product described in this manual and products for use with it, are subject to continuous development and improvement. All information of a technical nature and particulars of the product and its use (including the information and particulars in this manual) are given by Acorn Computers in good faith. However, it is acknowledged that there may be errors or omissions in this manual. A list of details of any amendments or revisions to this manual can be obtained upon request from Acorn Computers Technical Enquiries. Acorn Computers welcome comments and suggestions relating to the product and this manual.

All correspondence should be addressed to:-

Technical Enquiries
Acorn Computers Limited
Fulbourn Road
Cherry Hinton
Cambridge
CB1 4JN

All maintenance and service on the product must be carried out by Acorn Computers authorised dealers. Acorn Computers can accept no liability whatsoever for any loss or damage caused by service or maintenance by unauthorised personnel. This manual is intended only to assist the reader in the use of the product, and therefore Acorn Computers shall not be liable for any loss or damage whatsoever arising from the use of any information or particulars in, or any error or omission in, this manual, or any incorrect use of the product.

WARNING: Systems 2, 3, 4 and 5 and the BBC Microcomputer must be earthed.

IMPORTANT: The wires in the mains lead for the system 5 computer and the BBC Microcomputer are coloured in accordance with the following code:

Green and Yellow	- Earth
Blue	- Neutral
Brown	- Live

If the colours of the wires in the mains lead of this apparatus do not correspond with the coloured markings identifying the terminals in your plug, proceed as follows:

The wire which is coloured green and yellow must be connected to the terminal in the plug which is marked by the letter E, or by the safety earth symbol ⏚ , or coloured green or green and yellow. The wire which is coloured blue must be connected to the terminal which is marked with the letter N, or coloured black. The wire which is coloured brown must be connected to the terminal which is marked with the letter L, or coloured red.

In the case of apparatus supplied with a moulded plug , please note the following:

If the socket outlet available is not suitable for the plug supplied, the plug should be cut off and the appropriate plug fitted and wired as previously noted. The moulded plug which was cut off should be disposed of as it would be a potential shock hazard if it were to be plugged in with the cut off end of the mains cord exposed. The moulded plug must be used with the fuse and the fuse carrier firmly in place. The fuse carrier is of the same basic colour* as the coloured insert in the base of the plug. Different manufacturers' plugs and fuse arriers are not interchangeable. In the event of loss of the fuse carrier, the moulded plug MUST NOT be used. Either replace the moulded plug with another conventional plug wired as previously described, or obtain a replacement fuse carrier from an authorised Acorn or BBC Microcomputer dealer.

* not necessarily the same shade of that colour

IMPORTANT: The wires in the mains lead for the mains adaptor for the Acorn Atom or the Econet clock or the Econet Terminator accesories are coloured in accordance with the following code:

Blue -Neutral

Brown -Live

As the colours of the wires may not correspond with the coloured markings identifying the terminals in your plug, proceed as follows:

The wire which is coloured blue must be connected to the terminal which is marked with the letter N, or coloured black. The wire which is coloured brown must be connected to the terminal which is marked with the letter L, or coloured red.

IMPORTANT: In the event of the fuse blowing, it should be replaced, after clearing any fault, with a 3 Amp fuse that is ASTA approved to BS 1362.

Introduction

The ECONET is a low cost local area network capable of linking many computers together so that they can share information and resources. It has been designed for use with all Acorn Computers, and forms a valuable extension to the BBC Microcomputer System. In addition to low-level communication, the Econet provides a disc filing system and a printing service which can be shared by all machines attatched to the network.

This manual describes the user commands and facilities provided by the network itself, the Acorn File Server and the Acorn Printer Servers. The early chapters are intended to help you understand the basics and start using the network as soon as possible. More technical information about the network and how it should be installed and maintained is provided in later chapters.

In addition to this manual, the following documents are available:

The Econet Reference Card - a summary of network user facilities
The BBC Machine Printer Server Manual
The Model B File Server Manual
The Econet Technical Manual
The Econet Installation Leaflet

The information in the Econet Installation Leaflet is also contained in Chapter 8 of this manual, and should be read before attempting to install an Econet System.

Notes of errors and suggestions for improvement will be gratefully received. Comments should be sent to:

Technical Documentation Department
Acorn Computers Ltd
Fulbourn Road
Cherry Hinton
Cambridge
CB1 4JN

1 Getting going

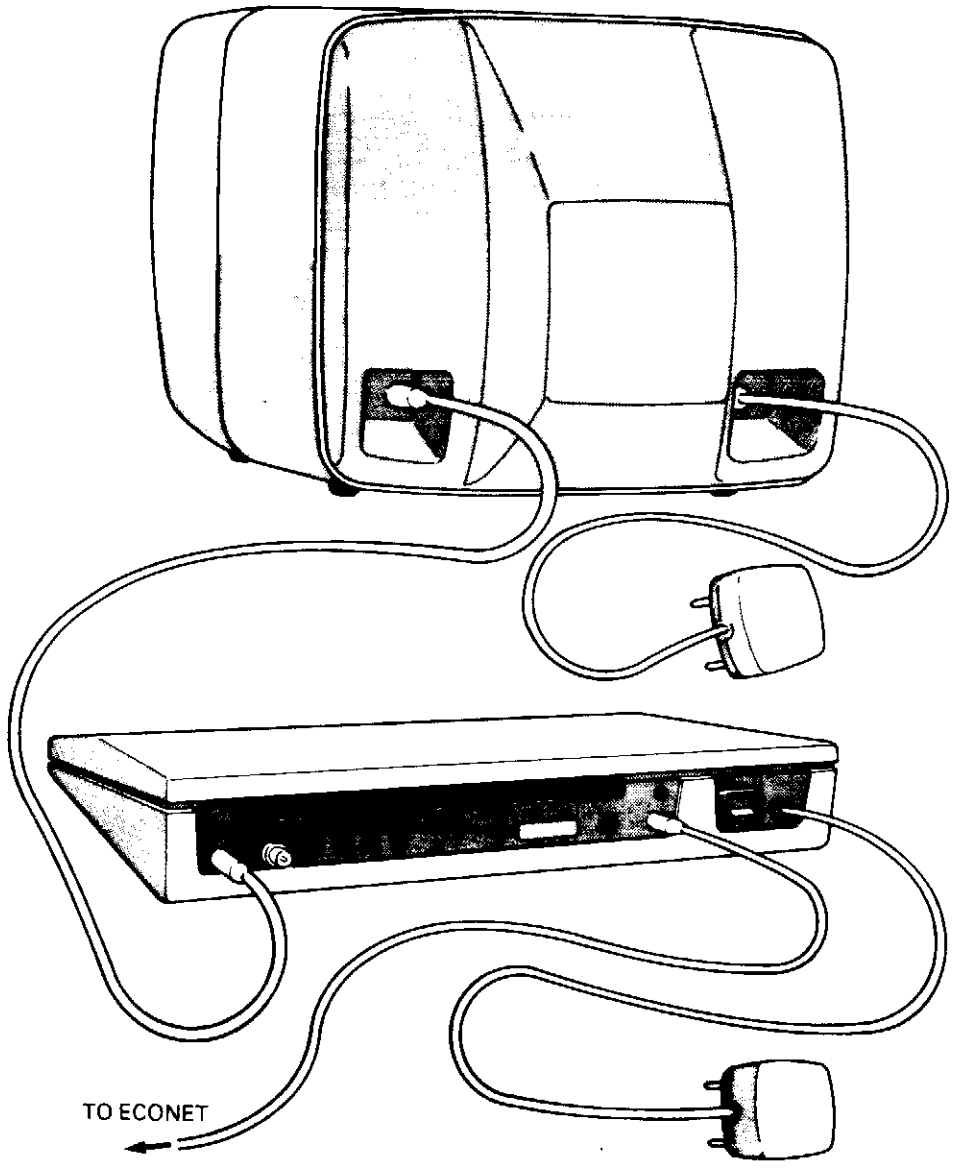
To get plugged into the Econet network you must have a computer suitably equipped as a user-station. At the moment, three types of computer can be connected to the network as a user-station. These are the BBC Microcomputer, the Acorn ATOM, and the Acorn Systems 3, 4 and 5. Whichever computer you use, it must be modified before using it with the Econet system. You may buy your computer already modifed (BBC Microcomputer models ANA 02, ANB 02 and ANB 04 are supplied complete with an Econet interface) or you may have it up-graded by your dealer. In either case, a computer for use as an Econet station must have the following:

Econet interface hardware.
A 5 pin din connector.
A copy of the Econet Version III software.
(Usually provided in an 8K EPROM for the BBC Microcomputer or a 4K EPROM for the Atom).

To get going you will need a monitor or television to use with the computer. Plug in and switch on the computer, plug the TV or monitor into the computer and switch that on also. Plug the 5 pin DIN connector from the computer into an Econet socket. The pictures overleaf may help you get these bits and pieces sorted out.

Obviously, you will need a network to plug in your station. Details on how to install a network are given in Chapter 8.

Figure 1.1 Plugging in a user station



1.1 Initialising the ECONET software

Once you have successfully connected your computer to the electrical supply and the network socket, it is necessary to start the ECONET software. This is done differently for each of the three types of user station so the next three sections tell you what to do for each machine. In addition, each machine can be started in a number of ways.

1.1.1 - On the BBC Microcomputer

If on power-up either of the two following messages appears, then your computer has already automatically selected the Econet filing system.

```
BBC Computer 32K

Econet station nnn

BASIC
>
```

or

```
BBC Computer 32K

Econet station nnn  No Clock

BASIC
>
```

The second is the 'No Clock' message. It is not necessarily a serious problem. Section 1.1.4 tells you what to do.

If neither of these messages appears, then you will have to select the Econet system. This you can do by typing 'N' BREAK, holding both keys down together, or CTRL 'N' BREAK, holding all three keys down together. Alternatively you can type

```
*NET
```

If you use one of the methods involving resetting the machine (pressing the BREAK key), you should obtain one of the two messages shown above. If you use the *NET command no special message will appear.

NOTE- WELCOME TAPE
If you wish to use the cassette filing system, for example to use the WELCOME tape, you must first do one of the following:

Either type

```
*TAPE
```

or press SPACE BREAK (holding both keys down together) or press DELETE

BREAK (holding both keys down together). You can now continue as on page 11 of the BBC Microcomputer User Guide. If you later want to use any of the facilities of the Econet system you will have to select it by one of the methods described above. This procedure is necessary because the BBC microcomputer can only deal with one filing system at a time and both the Econet and cassette systems are filing systems (*TAPE and *NET select different filing systems - see Chapter 9).

1.1.2 - On the Atom

When you think you have connected everything together properly press BREAK on the keyboard of your computer. If everything is working you should get a message on the screen which looks something like this:-

```
ACORN  ATOM  ECONET   3.1A                    235
>                       |                     |
                        |                     |
                Release/Version Number        |
                                        Station number
```

If all is not quite well you might get this message:

```
ACORN  ATOM  ECONET   3.1A  NOCLK             235
>                             |
                              |
                      Stands for 'No Clock'
```

Section 1.1.4 tells you what to do if you get this 'No Clock' message.

1.1.3 - On the Acorn Systems 3, 4 and 5

The System computers will not initialise the ECONET when you press BREAK. You need to make the computer jump to the first line of executable code in the ECONET software. This is accomplished by typing:

```
GO E003        (Starts the software allowing communication over the
               network.)
```

After typing this you should get the message

```
Econet 3.10
NOS V.IIIA
```

You may probably also want to type one of the following:

```
GO E000        (Enables just the network and returns "Econet 3.10")
GO E009        (Enables the local disc filing system in the station and
               the network printer)
```

1.1.4 The 'No Clock' message

This means that although your computer is switched on and contains the necessary hardware to connect to the network, a valid 'clock signal' cannot be detected. There are two major reasons why you might get this problem. Number one is that you might have forgotten to plug your computer into the network with the 5 pin DIN plug. Easily remedied - plug it in and start again. Failing that, move on to major reason number two. There is no clock generator connected to the network to generate the clock signal. (One and only one clock generator must be connected to the network to do this). Seek out the person in charge of your network and ask for a 'clock signal'. Probably you will not be the first person to ask as the absence of a clock signal will prevent everyone from using the network.

1.1.5 What next?

Let us assume that you managed to connect to the network without a hitch. The cursor will be positioned directly after the '>' prompt for BASIC. You can now enter a program or type commands to make use of the network facilities. The most useful facility now available to you is the File Server. Chapter 2 tells you all about it.

2 The file server

2.1 What is the file server?

The file server is a machine code program written to run on an Acorn System 3, 4 or 5 computer or the BBC Microcomputer with a second 6502 processor and a disc system. It provides a disc file system which can be shared by the stations connected to the network. Common access to the disc drive(s) allows any authorised user to store and retrieve information.

The system supports a hierarchical directory structure, each user has as many directories as he requires. Simple protection facilities are provided to ensure that one user's files can be protected from both other users and himself.

2.1.1 Purpose of this chapter

This chapter gives a complete description of the facilities provided by the file server. It starts with the simplest form of use and introduces more complex operations in later sections. It is intended to be an instructional section rather than a reference manual or technical description. Accordingly, lists of error messages, syntax definitions etc. have been confined to later chapters.

To find out how to use the file server simply to load and save files, it is suggested that you start at section 2.2, which gives a simple introduction to using files over the network. Section 2.3 introduces the idea of multiple directories, and further sections gradually deal with more complex features such as file security and library commands.

Chapter 7 describes how to start and maintain the file server itself. This chapter is provided for the people responsible for running the file server, making sure the correct discs are available, and generally maintaining the file server for other users.

Readers who are familiar with using filing systems and in particular hierarchical systems such as UNIX or TRIPOS may find that Appendix A gives a general introduction to the system sufficient to allow them to bypass some of the earlier sections.

Details of how to use other facilities of the ECONET such as printer servers, the NOTIFY command etc. are covered in chapters 4 and 5.

2.1.2 Not starting the file server!

The person in charge of the network usually starts the file server, so you don't have to. You can immediately start using the file server and if you are a 'registered' user you can create files and directories of your own. If you ARE the person in charge of the network and you want to find out how to start the file server, read chapter 7.

2.2 Simple File Commands

2.2.1 A single directory

Although the file server supports a multi-directory file structure, it can be used in its simplest form just to store a list of files. This section describes the use of simple commands to save, load and inspect files in a single directory.

2.2.2 Commands

File server commands are identified by preceding them with a "*" character. This distinguishes them from, for example, BASIC commands, PASCAL commands, or commands relating to a particular application program (e.g. an editor or assembler).

Commands can be used in BASIC programs (see section 5.2 for an example) and can be shortened by the "." character (see appendix B for abbreviations).

If the File Server does not recognise a command, the message "Bad command" will be displayed and an error will occur.

2.2.3 Identifying Yourself

Before the file server can deal with file commands being sent to it from a station on the ECONET, it must know who is using that station. So the first thing to do before using files on the ECONET is to identify yourself.

Each user has an identifier. Each identifier can be up to ten characters in length and must start with a letter. The user must remember his identifier which he sends to the file server using the *I AM command. e.g.

```
*I AM ALEXANDER
```

ALEXANDER is now in communication with the file server.

This process of identification is known as "logging on" and will be refered to at various places in this manual.

2.2.4 SAVE

SAVE is a BASIC command which saves the current BASIC program text as a file. e.g.

```
SAVE "MYPROG"
```

saves the current BASIC program as a file which belongs to the user who is logged on at this station.

2.2.5 *SAVE

*SAVE saves an area of memory as a file. e.g.

```
*SAVE MYDATA 3000+500
```

saves the area of memory from 3000 to 3500 as a file called MYDATA, owned by this station. In this case 3000 was the start address of the area in memory, 500 was its length. An alternative format for the command is:

```
*SAVE MYDATA 3000 3500
```

where 3000 is the start address as before, but the second parameter is the finish address of the section of memory you want to save, in this case 3500. This command is useful for saving machine code programs or data.

It is also possible to specify a third parameter to a *SAVE. This is the address where execution will start if the file is loaded as an executable command (see section 2.6.1). e.g.

```
*SAVE BASIC C000+1000 C2B2
```

saves a machine code file which will be executed at C2B2 if loaded as a command.

The execution address can be left out of the command, in which case it defaults to the load address.

A fourth parameter may be specified which will be the address at which the file is loaded back (if this is to be different to the start address). For example:

```
*SAVE PROG 3000 3500 5050 5000
```

will save a block of memory from 3000 to 3500. If the file is loaded it will be loaded at 5000 and the execution address if it is loaded as a command will be 5050. The load address parameter is optional. If it is to be specified, the execution address must also be explicitly given when the file is saved.

2.2.6 LOAD

LOAD is a BASIC command which loads a file as a BASIC program. e.g.

```
LOAD "MATHS"
```

loads a BASIC program called MATHS.

2.2.7 *LOAD

*LOAD loads a file into a particular area of memory. e.g.

```
*LOAD MYDATA
```

will load the file MYDATA at the load address of that file. This address is either the start address given when the file was saved, or the load address if this was specified (the fourth parameter of the *SAVE command).

```
*LOAD MYDATA 3200
```

will load MYDATA at location 3200.

2.2.8 Listing Your Directory - *CAT

The *CAT command provides a list (catalogue) of the files you have saved. A list (or set) of files in a filing system is known as a DIRECTORY, and each file is known as an ENTRY in the directory. (It is also possible to create directories as directory entries. This is dealt with in section 2.3).

After the first three lines, the files are listed in alphabetic order, one per line with the access rights (see section 2.2.13) following each name. e.g.

```
MATHS      LR/
MYPROG     WR/R
```

If no files are listed, the directory is empty (you have not saved any files).

The first three lines contain information which is explained in detail later in this manual. The top line starts with the name of the directory being displayed. After this is the cycle number (which changes each time an entry is deleted or created) and then the word 'Owner' or 'Public' according to your 'access rights' to this directory. The second line has the disc title and the auto-start option (see section 2.8). The third line shows the name of your currently selected directory and the name of the directory currently selected as the library.

```
DIRNAME      (33)    Owner
THEDISCTITLE         Option 0  (Off)
Dir. CSDNAME         Lib. LIBNAME

MATHS   LR/
MYPROG  WR/R
```

The meaning of ownership of a directory is dealt with in detail in section 2.4, but in the simple case you will always own your directory.

2.2.9 File Information - *INFO

*INFO prints a list of information about a particular file. e.g.

```
*INFO MATHS
```

produces a string of information about the file MATHS. The format is:

<name> <access> <load address> <execute address> <size> <date - dd:mm:yy> <System Internal Name>

The information may be split onto more than one line if the screen mode in use does not allow enough characters across the screen. For example in mode 7 (on the BBC computer) the information for a typical BASIC program might be:

```
MATHS   WR/R      FFFF1200  FFFF801F
00044A  16:09:82  000450
```

The date is the date the file was last saved, and the System Internal Name (SIN) is the start sector of the file on the disc.

2.2.10 *EX/*INF

This is a program held in the Econet ROM on the BBC Microcomputer (*EX) and on the file server disc for ATOMs and Systems (*INF). The command will cause the computer to do a *INFO for all the files in the specified or current directory. A typical listing might be:

```
DIRNAME         (33)      Owner
THEDISCTITLE              Option 0 (Off)
Dir. CSDNAME              Lib. LIBNAME

MATHS     LR/       FFFF1200   FFFF801F
00044A    21:7:82   000450
MYPROG    LR/R      FFFF2900   FFFF2900
000351    13:7:82   00005D
MYDATA    LR/       FFFF1800   FFFF801F
000500    15:7:82   000043
PASCAL    DL/       00000000   00000000
000200    10:5:81   000075
```

where PASCAL is a directory (see section 2.3)

2.2.11 *DELETE

Simply deletes a file from the directory. e.g

```
*DELETE MATHS
```

Once deleted a file cannot be restored. Locked files cannot be deleted and the message 'Entry locked' is displayed if you try. You can only delete files in directories which you own.

2.2.12 *RENAME

This command changes the name of a file. For example:

```
*RENAME OLDFILE NEWFILE
```

changes the name of file "OLDFILE" to "NEWFILE". You can only rename files (not directories), you can't rename locked files and you can't rename across discs. You can rename a file and change its directory (provided you have sufficient access). For example you can use something like:

```
*RENAME A.OLDNAME B.NEWNAME
```

which will move file "OLDNAME" in directory A and place it in directory B under the name "NEWNAME".

2.2.13 File Access - *ACCESS

Many people will use a single file server, and you may want to protect important files from both other users and from accidents such as saving a data file into a program file or deleting an important program.

It is therefore possible to set the ACCESS STRING for a file. e.g.

```
*ACCESS MATHS LWR/WR
```

Each letter in the access string refers to a type of access to the file:

```
L - locked => the file cannot be deleted or overwritten.
W - write  => the file can be written to
R - read   => the file can be read.
```

The "/" character separates the OWNER'S access (on the left) from PUBLIC access (on the right). So in the example, MATHS can be read and updated by both the owner and anybody else using the system.

If any character is not present in the string, the access it represents is taken to be unset. So:

```
*ACCESS MATHS LR/
```

sets MATHS to be locked and readable only by the owner.

```
*ACCESS MATHS
```

sets MATHS to be unlocked and accessible by nobody.

When a file is saved, the access is set to a default of WR/.

Owner and public access is described more fully in section 2.4, but for simple use it is safe to assume that you own the files you have saved, and everybody else has only public access to them.

NOTE
When a file is saved, the file server acts as though a new file is being created, and the old one, if present, is deleted. Thus you should protect against accidental saving with the LOCK character, not the WRITE character. Other people can never save into your files, whether they are locked or not (see section 2.4), unless they are system privileged (see section 7.7).

The WRITE character refers to random access files (see section 2.10).

2.2.14 A Note on File Titles

File titles must be at least one character and up to ten characters

and can include upper and lower case alphabetic characters, numerics and the following punctuation symbols: "." ":" "!" "-".

NOTE
"." and ":" have special meanings explained in section 2.3.
Upper and lower case alphabetic characters are not distinguished. Thus "MATHS", "MaThS" and "maths" are all the same file name.

2.2.15 Multiple File Servers

All file server commands are addressed to a station number on the ECONET by the ECONET software. This station number is stored by the operating system, and is set by default to 254.

If you wish to log on to a file server with a different number, type:

```
*I AM 234 JOEY
```

where 234 is the station number of another file server.

From that point on, all file server commands will be sent to station 234, unless you press CTRL and BREAK together. (Just BREAK on the Atom and System computers). When this happens, you remain logged on, but the station number reverts to 254, and it is neccessary to reset it, either by logging on again, or by directly changing the file server number at its location in your stations memory. On the BBC machine this is done by calling the appropriate OSWORD routine, on Atoms and Systems by poking the relevant memory locations. Full details are provided in Chapters 11 and 12. A soft reset (just BREAK) on the BBC Microcomputer does not reset the number of the selected file server.

The ECONET can easily support several file servers, which have different numbers. They operate quite independently of each other, and one user can be logged on from one station to as many as required. This requires a knowledge of the operating system interface routines (Chapters 11 and 12) and some understanding of the file server interface (Chapter 13). In particular it should be noted that the three user environment handles returned by each file server are unique to the file server that returned them. Thus when the file server number is changed, these handles must also be changed to the ones produced by the new file server.

A single user can be logged on to the same file server from any number of stations, as the file server identifies people by their user identifier, not station number. The only limitation is the ability of the user to remember where he is logged on to what.

2.2.16 Monitor messages

The command *OPT1 is used to set the "message switch" which controls the details of a message displayed whenever a file is loaded or saved.

*OPT1,0 implies no messages are issued.
*OPT1,1 gives detailed information including load and execution addresses and length.

2.2.17 Summary

So far you have learnt how to use a single directory, save, load, delete and protect your files on it. In doing all this you may have come across some of the systems error messages. These are produced when the file server cannot understand your commands or something on the network is not working properly. All these errors are explained in Chapter 6. Common ones are NOT LISTENING, NO REPLY.

2.3. Creating and Using a Hierarchy of Directories

2.3.1 Directories Within Directories

In addition to containing files, a directory may have entries which are themselves directories. This allows any file owner to structure the way in which he keeps his files.

For example, the directory ALEXANDER, in addition to the files mentioned above, could have several directory entries corresponding to several different types of file. Pascal programs could be stored in a directory called PASCAL, and text-edited letters in a directory called LETTERS. The structure can be extended by, for example, creating further directories in LETTERS called MUM, DAD and BUSINESS to keep letters sent to various people.

The only limit on the number of directories the system can handle is the amount of space on the disc, so each user can create as large and complex a structure as he thinks he needs to cope with his files.

This type of structure, in which a directory can point to a number of directories "below" it, is called a HIERARCHICAL structure or tree structure. (see figure 2.1)

Throughout this section we will continue to use examples assuming that we are logged on as ALEXANDER, and are using the directory ALEXANDER.

2.3.2 Creating Directories

The *CDIR command creates a directory entry in the same way that *SAVE creates a file entry. So:

```
*CDIR PASCAL
```

creates the directory PASCAL as an entry in ALEXANDER.

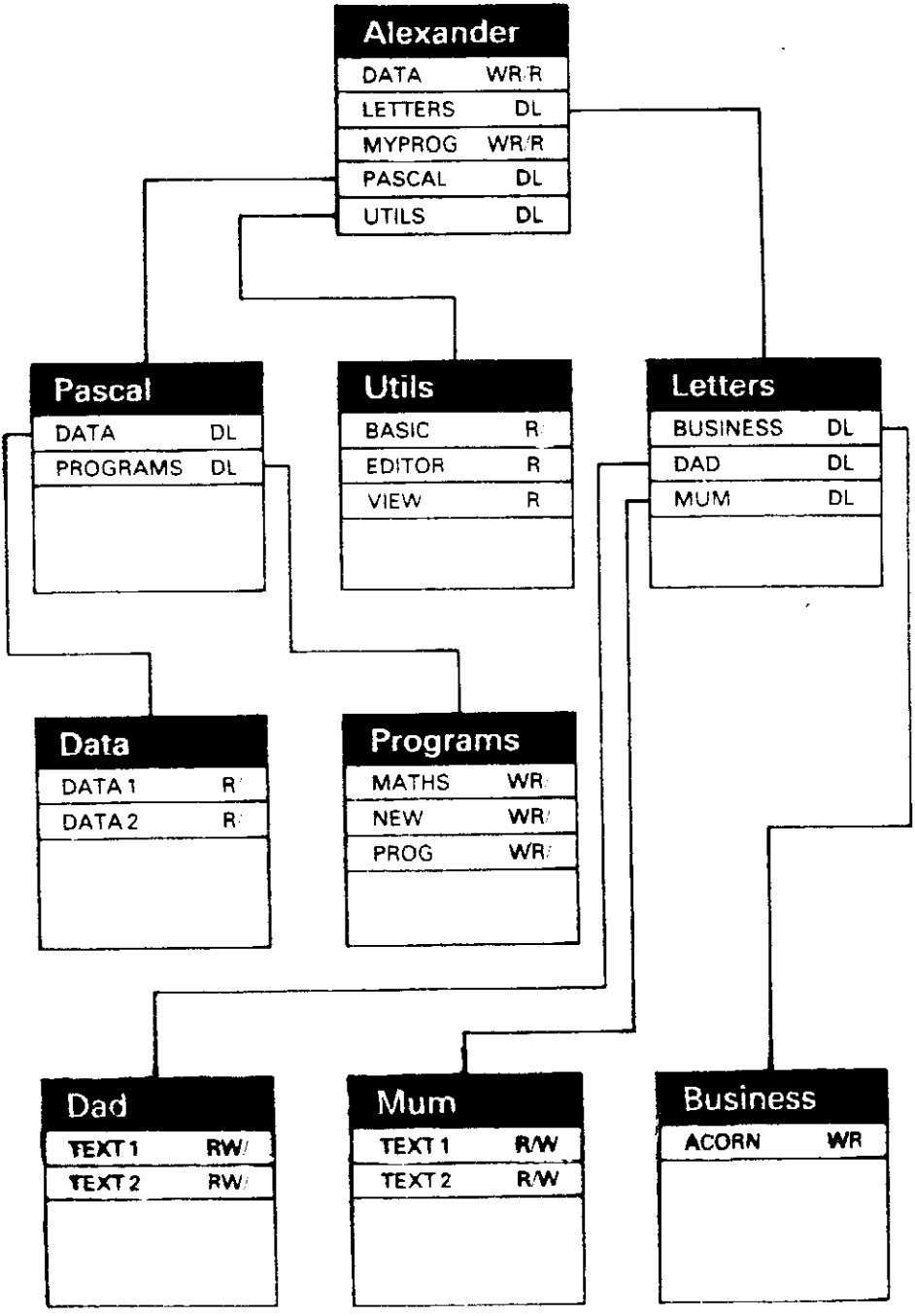
Executing *CAT will now list PASCAL as an entry among the files:

```
PASCAL    DL/
```

The D before the access string indicates that it is a directory, and that its default access is simply "locked". W and R do not have any meaning for a directory.

Executing *INFO PASCAL gives information very similar to file information, except that the load and execute addresses, which again have no meaning, are zeros.

Figure 2.1 A hierarcical file structure



2.3.3 File Names

In order to use files from directories other than the current one, directory names and file names can be strung together to make one composite file name. The sections of the name are separated by the "." character and each section corresponds to a level (directory) of the hierarchy. (The only limit to the length of this string of names is the size of the input buffer).

For example, with ALEXANDER selected, to access the file STATS1 in directory PASCAL, the title PASCAL.STATS1 is used. The section PASCAL is looked up in ALEXANDER, is found to be a directory, and STATS1 is then looked up in PASCAL. So the command:

```
*SAVE PASCAL.STATS1 3000 +34
```

saves a file into the directory PASCAL.

The command

```
*CAT PASCAL
```

now lists the directory PASCAL (and shows STATS1 as an entry), and

```
*INFO PASCAL.STATS1
```

gives us information about the file STATS1 in PASCAL.

Similarly, to load a particular business letter:

```
*LOAD LETTERS.BUSINESS.TOACORN
```

Each section of a long file title is subject to the rules described in section 2.2.14 (10 characters only etc.).

2.3.4 Selecting Directories

In the examples so far, each time we quoted a file name (in SAVE, *CAT etc.), it was looked up in the directory ALEXANDER. This would be inconvenient if we wished to do some work using the PASCAL files, and so it is possible to change the directory in which file names are looked up. The command used is *DIR (select directory). e.g

```
*DIR PASCAL
```

selects PASCAL as the CURRENTLY SELECTED DIRECTORY (CSD) in which all file names are now looked up. So, the command

```
*LOAD STATS1
```

loads the Pascal file STATS1 from the directory PASCAL.

Similarly, *CAT now produces a list of entries in PASCAL, not ALEXANDER. The first line of *CAT also displays the last section of the current directory name, which is now PASCAL.

To return to the directory which was selected when ALEXANDER logged

on, the command *DIR with no file name can be used.

2.3.5 The ROOT Directory and USERS ROOT Directory

So far we have only been concerned with a single user's filing system stemming from a single directory, in this case ALEXANDER. In fact, a disc is itself a large file hierarchy with many users each having a personal sub-system such as the one described above.

At the top of the system is a directory called the ROOT which contains a number of directories such as ALEXANDER. Each of these is a USERS ROOT DIRECTORY (URD) and forms the basis of a user's personal file structure. The ROOT is referred to using the character "$" and can be incorporated in file titles like any other directory.

This now makes it possible to look at and use other users' files (providing you have access - see section 2.4). For example:

```
*CAT $.JOHN
```

gives a catalogue of the directory JOHN in the root directory. This directory is not part of ALEXANDER's system and he only has public access so, for example, he cannot save files or create directories into it.

A file title using the root is like any other file title and can therefore be used with any command. e.g.

```
*DIR $.JOHN
```

selects JOHN's directory as our current one (although we still don't have access to his files).

```
*LOAD $.JOHN.BASIC.PROG1
```

loads one of JOHN's BASIC files. This would only be allowed if JOHN had allowed us READ access to his file.

```
*CAT $
```

gives a list of the files and directories available in the root directory.

The root is created and put on the disc when the disc is initialised - see chapter 7.

2.3.6 Selecting Discs

A machine running the file server can support several disc drives. Each disc contains a complete filing system as described above, with a root at the top. Each person using the file server has a CURRENTLY SELECTED DISC, which determines which file system he is using.

The currently selected disc name is displayed in the second line of a catalogue. The file server selects a disc for a user when he logs on by searching each disc on the system for a directory in the root which is the same as the user identifier. When such a directory is found, that disc becomes the user's currently selected disc, and the

directory becomes both his users root directory and CSD. If no such directory is found, the disc in the left hand drive becomes the users current disc, and the root of that disc his URD and CSD.

To select a different disc, the *SDISC command is used. Discs are identified by name, which is simply quoted after the command:

```
*SDISC "FORM-THREE-MATHS"
```

If the disc is found, the same procedure for selecting URD and CSD is used as for logging on, and all file operations are interpreted in the new filing system.

It is also possible to temporarily access files on another disc, by quoting the disc name before a file title. (A disc name is distinguished in a file title by preceding it with the ":" character). e.g.

```
*CAT ":MATHS-DISC.JOEY.DATA"
```

gives a catalogue of a directory located on a different disc. Once again, this is a general facility, so the following commands are valid:

```
*DIR ":MATHS-DISC.JOEY"

LOAD ":FORM-THREE-MATHS.JOEY.STATS"
```

NOTE
The root does not have to be specified when quoting a disc name (except when dealing with the root explicitly) so the above both refer to $.JOEY on the respective discs. To catalogue the root itself on FORM-THREE-MATHS:

```
*CAT ":FORM-THREE-MATHS.$"
```

2.3.7 Deleting Directories

Directories can be deleted using the *DELETE command, but ONLY if they themselves do not contain any entries. If you wish to delete a directory and its entire contents, this must be done entry by entry.

2.4. Owner and Public Access

2.4.1 What's the difference?

As we have seen, the owner of a directory may create entries in that directory (files or further directories), set the access to entries and has his own access governed by the "owner" field of any access string.

If a user is using a directory which he does not own, he has PUBLIC access. This prevents him from creating entries (by saving or *CDIR) and limits him to the "public" field of each access string. Also, he cannot set the access string of any entry.

2.4.2 Becoming the Owner

If, when a user logs on, a directory is found in the root which matches his user identifier, he is given OWNER access to that directory, and any directories which are entries in that directory (it also becomes his CSD and URD - see 2.3.6). He is therefore owner of his complete directory system.

However, he only has PUBLIC access to any directories or files outside that system, and cannot save files into somebody elses directory.

e.g. After logging on as ALEXANDER and executing:

```
*CAT $.JOHN
```

the first line of the catalogue will contain the word "Public" after the directory name indicating that ALEXANDER only has public access to that directory.

It should be noted that if ALEXANDER refers to his own files through the root, he will be treated like anybody else and given public access only. So:

```
*CAT $.ALEXANDER
```

will also only display public access, whereas *CAT with ALEXANDER selected as the users root directory will display owner access.

To emphasise the point:

```
*DIR $.ALEXANDER
```

selects $.ALEXANDER as the CSD, but only with public access, whereas:

```
*DIR
```

selects the same directory, but with owner access, since *DIR requests a return to the User Root Directory.

If there is no directory which matches the user identifier, the user is given public access only to the whole disc.

This set of rules also applies when selecting a disc so that users' files on one disc are protected from people using a separate disc on the same system.

2.5 Passwords

2.5.1 How they are used

To prevent anybody who knows your user identifier from logging on as you and therefore gaining owner access to your files, it is possible to set a PASSWORD.

This is a string of up to 6 letters and numbers which should be known only to you and which must be quoted when you log on. e.g.

```
*I AM ALEXANDER JULIE
```

will allow ALEXANDER to log on if JULIE is his password. But:

```
*I AM ALEXANDER
```

and

```
*I AM ALEXANDER CRUST
```

will produce the error "Wrong password".

Note
ON ATOMs and Systems you can prevent your password appearing on the screen by typing CTRL U to turn off the screen, then type your password and then type CTRL F to turn the screen on again. This is not possible on the BBC Microcomputer and you are advised to clear the screen after typing your password to prevent other people seeing it.

2.5.2 Setting the Password - *PASS

To set a new password, type:

```
*PASS <old pw> <new pw>
```

For example:

```
*PASS "" JULIE
```

sets ALEXANDER's password to JULIE, and

```
*PASS JULIE MOZART
```

sets it to MOZART.

Also:

```
*PASS MOZART
```

removes ALEXANDER's password completely. Give careful thought to your choice of password if you want it to be effective. Your girl-friend's name or your car registration are likely guesses for anyone trying to break into your files.

2.5.3 The Password File

The file server checks passwords by looking them up in a special file called the PASSWORD FILE. This exists on a disc selected by the file server when it is started up (see chapter 7).

It is important to note that a password is therefore associated with a disc, and that if that disc is not in the system when a user tries to log on, he will be looked up in whatever password file is in use by the system.

If a user tries to log on when no password file exits on any of the discs on the system, the error "Password file not found" will occur.

Making sure that a suitable password file exists is part of the job of operating and maintaining the file server, described in chapter 7.

2.6 User Commands and Libraries

2.6.1 Introduction - Commands

It is possible to load and execute machine code programs in a single step simply by typing their name after the "*" character indicating a file server command. e.g.

```
*PASCAL
```

would load the file PASCAL from the current directory, and then jump to its execution address automatically, in this case starting the BBC Microcomputer Pascal interpreter program.

If the name PASCAL had not been found in the currently selected directory, the file server would have then searched the currently selected LIBRARY.

2.6.2 Libraries

A library is simply a directory which is searched to look for command names not recognised by the file server program. When a user logs on, the file server will try to select the directory $.LIBRARY as the user's current library. If it cannot find $.LIBRARY, the root will become the selected library.

A different library can be selected using the command *LIB. e.g.

```
*LIB $.ALEXANDER.MYLIB
```

This command means that any command which the file server does not recognise is first searched for in the CSD and secondly in $.ALEXANDER.MYLIB. Only when it is not found in the library will the error "Bad command" occur.

For example, typing:

```
*EDIT
```

means that a file called EDIT is looked for in the CSD, and if not found there, in $.ALEXANDER.MYLIB. If EDIT is found, it will be loaded and executed. Note that it is impossible to load and execute a directory, so an error would occur if a directory called EDIT existed in either of those directories.

The library allows all users of the system to share useful application and utility programs easily. In particular, if command programs are kept in $.LIBRARY, using them is simply a matter of logging on and typing the program name as if executing any other file server command.

2.6.3 *RUN and */

Machine code programs can be loaded and RUN using either of these commands. For example, typing:

```
*RUN EDIT
```

or

```
*/ EDIT
```

will cause the file EDIT to be loaded and run. The file is looked up in the currently selected directory and then the current library in the same way as loaded commands.

2.7 *EXEC and *SPOOL

These two commands are useful for dealing with BASIC files and commands. They are only available on the BBC Microcomputer.

2.7.1 *EXEC

This command reads byte by byte all the information in a named file as if it were being typed on the keyboard. This is useful when you find that you are repeatedly typing the same sequence of commands. Instead you can build up an "EXEC" file consisting of these commands and type *EXEC <filename> each time you want to use this sequence of commands. The argument can be a general file specification, that is, it can include directory name(s).

For example

```
*EXEC $.JOEY.COMM
```

will read the file COMM in the directory JOEY byte by byte as if entered at the keyboard.

2.7.2 *SPOOL

This command prepares a named file to receive the information subsequently displayed on the screen. This is useful for making an "EXEC" file (see section 2.7.1). For example, typing

```
*SPOOL FILE
```

will open a file called FILE on the current directory, assuming that the user has sufficient access to do this (see section 2.4), and anything subsequently displayed on the screen will also be sent to FILE. Typing

```
*SPOOL
```

turns off spooling and closes FILE. It is importatn that you turn the monitor messages off when spooling otherwise they may interfere with the data you are trying to send to the file.

NOTE
If you have a disc filing system attached to your computer, the disc system commands *BUILD, *LIST, *DUMP, *TYPE etc. are available - see the Disc Filing System manual for details.

2.8 Auto-start facilities

The facilities described in this section are available on the BBC Microcomputer only.

On the Econet system, auto-start may be selected to occur after either or both of the following actions

(1) Resetting the machine with CTRL BREAK or BREAK or on power-up, with the Econet system fitted.
(2) Logging on (as in section 2.2.3)

Whichever auto-start is chosen, a file called !BOOT will either be ignored, LOADed, RUN or EXECuted. The action on the file !BOOT is controlled by setting the users OPTION as described in the next section.

2.8.1 Setting option

The option for a particular user is selected by that user by typing

```
*OPT 4,n
```

where n is a number in the range 0 to 3. The option numbers have the following meaning:

```
OPTION 0 : ignores !BOOT
OPTION 1 : *LOAD's !BOOT into memory at its load address
OPTION 2 : *RUN's !BOOT as a machine code file
OPTION 3 : *EXEC's !BOOT
```

A users option can only be set by the user when logged on.

2.8.2 Auto-start when logging on

When a user logs on, a file called !BOOT is looked for in the users root directory. If the file is found, it is treated according to the option set for the user (see section 2.8.1). If there is no file called !BOOT in the users root directory, or if something goes wrong while loading, an appropriate error will occur (e.g. "Not found").

2.8.3 Auto-start when resetting the machine

If a BBC Machine is started by pressing SHIFT BREAK (both keys held down together), the Econet will attempt to auto-start without further intervention. It will do this by attempting to logon to the default file server as user BOOT. The action which is then taken depends on the option set for user BOOT and the contents of the !BOOT file in user BOOT's directory.

If BOOT does not exist, or has no !BOOT file, there will be no auto-start, an error will be generated and the machine will hang until reset with BREAK etc. It should be noted that the user BOOT is no different from any other user name, and can be created by a system user in the normal manner (see 7.7.2).

It is possible to configure a BBC Machine so that the action on BREAK and SHIFT BREAK are swopped. This has the advantage that the BBC Machine will always try and auto-start (even on power-on) unless the SHIFT key is pressed. This feature is controlled by link 5 of the set of 8 links located at the right-hand front of the keyboard. These are numbered 1 to 8 from left to right and link 5 must be made to enable auto-start without the SHIFT key. Note that these links are only read by the operating system on hard reset of the machine, so changing the link and then simply pressing BREAK will have no effect on the auto-start state.

2.9. Utility Programs

2.9.1 Introduction

The commands described in the following section are command programs which are loaded and executed from a library as described in section 2.6.

There are generally three versions of each command, for ACORN SYSTEMS, BBC Microcomputers and ATOMS. On the disc provided with the system, the ATOM versions will reside in $.ATOMLIB, the SYSTEM versions in $.SYSLIB and the BBC Microcomputer versions in the default library $.LIBRARY.

Atom and System versions of each program are loaded into the calling machine at &2800. BBC versions are loaded into space reserved by the Network filing system.

2.9.2 *INF

This command gives a list of all the entries in a directory and, in addition, provides information about each entry in a format similar to the *INFO command. The "page mode" feature of the ATOM may come in handy here. INF can only accept file titles of up to 12 characters. Titles longer than that will be truncated. This command corresponds to *EX in BBC machines. See also section 2.2.10.

2.9.3 *DISCS

This command will produce a list of the discs available in the current file server. The list will include the disc name and drive number.

2.9.4 *USERS

This command will produce a list of the users currently logged on to the network. The list will include the users name, the privilege and the station number.

2.9.5 Network Utilities

These programs, although loaded from the file server, do not concern the filing system itself, and are described in chapter 5.

They include:

*NOTIFY
*REMOTE
*VIEW
*PROT
*UNPROT

2.9.6 *PS

This command sets the number of the station which printed output will be sent to (the Printer Server number). e.g.

```
*PS 123
```

sets the Printer Server number to 123.

See section 4 for more information on printing over the network.

2.10 Random access

2.10.1 What is random access?

Random access is the name for reading or writing selected sections of a file. The disc systems available for the BBC, Atom and System computers all provide random access facilities. If you have written programs which make use of these random access facilities, they will run on the network without modification. The operating system primitives which provide these facilities are the same whether used with disc, cassette or over the network. The network filing system will intercept your program's input/output requests and re-direct these to the file server.

The concept of file handles (referred to as 'channels' on the BBC Microcomputer) is introduced here, followed by a general consideration of using random access in your BASIC programs. Assembler programmers should note that understanding the use of file handles (channels) is essential if you wish to directly address the file server.

2.10.2 Channels - Opening and Closing files

All the random access operations refer to files using a single byte number called a Channel ('Handle' on the Atoms and Systems). For example, the BGET# command in BBC BASIC takes a channel as an argument to indicate which file to read e.g.

```
340 nextletter%=BGET#C
```

This example is taken from the BBC Microcomputer user guide where all the relevant keywords are explained. In this case the letter C is the channel and the command will read the character in the file which is currently pointed to and assign its value to the variable 'nextletter%'.

The first step in using a file for random access is to assign a channel number to the file. Two commands are available in BBC BASIC to do this, OPENOUT and OPENIN. OPENIN is used for existing files which you want to read or update. A typical example of its use is:

```
230  X=OPENIN("cinemas")
```

This assigns the channel X to the file "cinemas" which must exist on the disc. After this line in your program you can use X as an argument in any of the read or write statements that you want to act on the file "cinemas".

OPENOUT is different because the named file is deleted and recreated with 1K bytes space reserved on the disc. If the named file is not found in the directory, a new one is created. This command is generally used to initially create a file for using with random access. Afterwards OPENIN would be used to open the file.

Note that files can only be opened for input and output if the user has sufficient access (see sections 2.2.13 and 2.4). The keyword OPENIN actually opens a file for update, so to use this keyword you need to have both read and write access. If you only have read access

to a file, and you wish to open it, you can use the operating system calls to open it for input only - see chapter 3.

The CLOSE# operation "deletes" a channel, so that the file server no longer recognises it and will give an error if used.

A single machine may have up to 8 valid handles at one time. However, three handles are taken up identifying the user environment (see chapter 13), so only 5 other objects can be opened at one time.

It should be noted that one user can open a single object several times and can therefore have several different handles refering to it.

2.10.3 Random access in BASIC

The general principles of random access are as follows:

The File must be given a handle (channel). This is a single-byte reference number used by the operating system to identify an area in RAM for loading sections of the file. On the BBC Microcomputer the words OPENIN and OPENOUT do this, FIN and FOUT on Atoms and Systems. Files can only be opened if the user has sufficient access.

The next task is to specify which section of the file you want to read or write. This is done by using a pointer to indicate a single byte in the file for the next read or write. Each file is considered as series of single bytes. In BBC BASIC the word used is PTR#. On Atoms and Systems it is PTR. It is used by assigning it a value, e.g.

```
PTR#<channel> = 1000
```

This means that the next byte to be read or written is number 1000 in the file.

There are a number of operators in both the BBC and Atom/System BASIC which allow you to read and write. These are:

BBC	**Atom/System**
INPUT#	GET
PRINT#	PUT
BGET#	BGET
BPUT#	BPUT
	SGET
	SPUT

Finally there is a function which tells you when the end of the file has been reached. (i.e. you have read or advanced the pointer to the last byte in the file). This is EOF# in BBC BASIC.

Later versions of BBC BASIC will have an extra keyword called OPENUP. This opens a file for update (reading and writing). OPENIN then becomes open for input only. Full explanation of how to use these BASIC keywords for random access is provided in the reference manual which comes with your computer.

Note
Multiple byte access using the operating system call OSBGBP (see section 3.4) is considerably faster than single byte access (as used

by BASIC). The operating system call OSBGBP should therefore be used where speed is important.

2.10.4 Using the Pointer

The file sequential pointer (PTR# or PTR) mentioned above can be used to select the next byte to be accessed or to extend the length of a file.

If you wish to read a particular byte of a file, all you have to do is assign the correct number to the variable PTR# or PTR. The next byte read will be from the position indicated by the value of this variable. The pointer is incremented by one after each byte is read.

If you wish to extend a file then all you need to do is to assign a value to PTR# (or PTR) which is beyond the end of the file. The variable EXT#, which usually gives the number of bytes in a file will give the total length of the file.

An example will probably serve to clarify this. Assume that the file "DATA" has been set up containing 20 bytes of information. If you open the file using

```
A=OPENIN"DATA"
```

then the variables will have the following values:

```
PTR#A  -   0
EXT#A  -  20
```

If you now type

```
PTR#A = 100
```

the end of the file will be moved to make the file 100 bytes long and the variables will have the following values:

```
PTR#A  -  100
EXT#A  -  100
```

and the file will be longer by 80 bytes. The extra 80 bytes will all be zeros.

Note
This can only be done if the file is opened for update (using OPENIN) otherwise you will get the "Outside file" error.
If you try to extend a file too far, you may get the error message "Disc full" if there is not sufficient space on the disc.

2.10.5 Interlocks

This is a very important feature included in the Econet system. Consider the following situation which might conceivably arise. You open a file to read using random access facilities. At the same time, somebody else opens the same file and starts to update it. It could easily happen that, between your reading one part of the file and another, the person updating the file changes some of the entries. This could produce disastrous results if some of the later data is

supposed to match up with some of the earlier data (for example a list of names and telephone number stored as a list of names followed by a list of corresponding numbers).

To prevent this situation arising, a system of interlocks is implemented. The system used is called "multiple reader, single writer" and is a common method of protecting multi-user systems. This works as follows:

> If a file has been opened for reading then anybody else can open the file for reading, but NOT for writing.
> If a file has been opened for writing, then nobody else (including yourself) can open the same file.

If you are not able to open a file for the above reasons you will get the error "Already Open".

Note
The BBC BASIC command OPENIN opens a file for updating, which includes writing. Thus a file can only be opened once using OPENIN.

3 Using the File Server from Assembler

This chapter describes the operating system interface for the filing system for the BBC Microcomputer. The interfaces for the ATOM and Systems are described in ATOMIC THEORY AND PRACTICE and the System DOS Manual respectively.

This chapter describes a series of operating system calls which can be used to open, close, read and write files.

Files are treated as a sequence of 8-bit bytes. Files can be accessed in one operation (using OSFILE) or a byte at a time (using OSBGET or OSBPUT).

Associated with each file are various parameters.

The load address specifies where a file should be loaded into memory if accessed using OSFILE.
The execution address is meaningful only if the file contains executable code, in which case this address represents an entry point if the file should be loaded and entered.
The extent of the file is its size in bytes (which may take the value zero).

In addition, associated with each open file is its sequential pointer. This represents the index (counting the beginning of the file as zero) of the next byte to be accessed.

A number of operating system calls are provided for dealing with files. In the following sections, A refers to the processor's accumulator, X and Y to the X and Y registers. The flags are referred to by the letters C, N, V and Z.

All addresses are in 6502 order, i.e. lo-byte first, hi-byte last.

3.1 OSFIND

Call address &FFCE (indirects through &021C).

Opens a file for reading/writing/update.

The value in A determines the type of operation.

(i) A non-zero

A<>0 causes a named file to be opened.
X (lo-byte) and Y (hi-byte) point to the name of the file to be opened. The name is represented by a string terminated by a carriage-return (ASCII &0D).

If A=&40 the file is opened for input only.
If A=&80 the file is opened for output only.
If A=&C0 the file is opened for reading and updating.

ON EXIT,
A<>0 implies that the file was successfully opened (A contains the handle for the opened file which must be provided when operating on the file).

A=0 implies that the file could not be opened.

C, N, V and Z become undefined.
The interrupt state is preserved. However, interrupts may be enabled during the operation.

(ii) A zero

A=&00 causes (a file)/(files) to be closed.

If Y<>0 the file whose handle is given by Y is closed.
If Y=0 all open files are closed (including SPOOL and EXEC files).

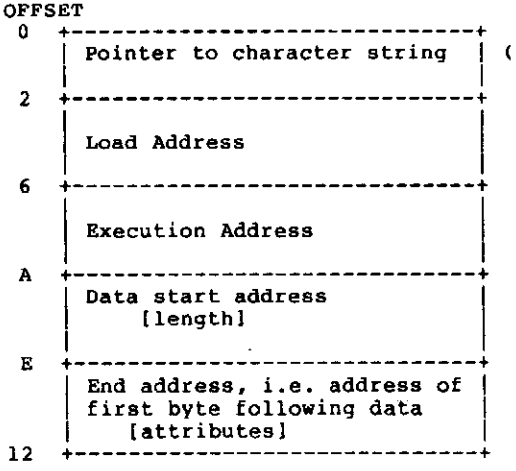
ON EXIT,
C, N, V and Z become undefined.
The interrupt state is preserved. However, interrupts may be enabled during the operation.

3.2 OSFILE

Call address &FFDD (indirects through &0212).

Load/Save data, Inquire/Alter catalogue information.

X and Y point to a control block (X lo-byte, Y hi byte):-



Notes: (i) The character string is terminated by &0D.

A contains a job description parameter.

&FF :- LOAD file to given address/file's address.
The low byte of the execution address (offset &06) determines whether the file should be loaded to it's own address (if non-zero) or to the supplied load address (if zero). The file's catalogue information will be written into the control block, see

item 5.
&00 :- SAVE data to file.
If start address = end address then the data is of zero length. The file's catalogue information will be written into the control block, see item 5.
&01 :- WRITE catalogue information for file.
Length information is not alterable.
&02 :- WRITE the load address to the catalogue information.
Only the Load address is required.
&03 :- WRITE the execution address to the catalogue information.
Only the execution address is required.
&04 :- WRITE the attributes to the catalogue information.
Only the attributes are required.
&05 :- READ file's catalogue information.
The Load address (offset &02), Execution address (offset &06), Length in bytes (offset &0A), Attributes (offset &0E) and Type (in A) are returned for the particular file.
&06 :- DELETE the file and return catalogue information. Returns 0 in A if object did not exist, otherwise non-zero.

The attribute of a file is a 4 byte (32 bit) item whose bits refer to the state of the protection flag, and the date. The bottom 8 bits have the following meanings:-

bit meaning
7 Undefined.
6 Undefined.
5 = 0 -> the file is not writeable by other users.
= 1 -> file is writeable by other users.
4 = 0 -> the file is not readable by other users.
= 1 -> the file is readable by other users.
3 = 0 -> the file is not locked.
= 1 -> the file is locked.
2 Undefined.
1 = 0 -> the file is not writeable by you.
= 1 -> the file is writeable by you.
0 = 0 -> the file is not readable by you.
= 1 -> the file is readable by you.

The Econet filing system places the date information in the next 16 bits using the following format:

Lo-byte - days
Hi-byte, bits 0 to 3 - months
Hi-byte, bits 4 to 7 - years since 1981.

The Type is a byte which describes the object which was found for READ:

Type 0 is no object found
Type 1 is file found
Type 2 is directory found

3.3 OSARGS

Call address &FFDA (indirects through &0214).

Reads/Writes an OPEN file's attributes.

X points to 4 locations in zero page.
Y contains the file handle.
A specifies the type of operation.
If Y is non-zero then OSARGS will do one of the following jobs on the file of which Y is the handle:

A=&00 read sequential pointer
A=&01 write sequential pointer
A=&02 read extent

The result being returned in the zero-page location pointed to by X. If the pointer is set to past the end of the file then the file will be padded such that intervening bytes read as nulls, A will be returned as 0 if the operation is performed, otherwise it will be left unchanged.

If Y is zero then OSARGS will do one of the following jobs on the filing system:

A=&00 Return type of filing system in A
0- No current filing system
1- 1200 baud CFS
2- 300 baud CFS
3- ROM filing system
4- Disc filing system
5- Econet filing system
6- Teletext/Prestel 'Telesoftware'

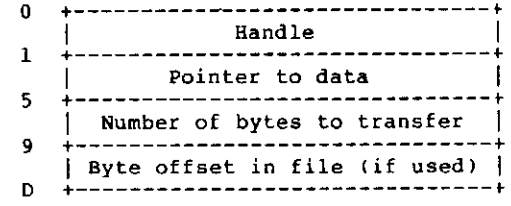
A=&01 Return address of rest of command line at X.

3.4 OSBGBP

Call address &FFD1

Write/read a group of bytes to/from a specified open file.

ON ENTRY,
X(lo-byte) and Y(hi-byte) point to an instruction block:



A determines the type of operation:
A=&01 Put byte using byte offset
A=&02 Put byte ignoring byte offset
A=&03 Get byte using byte offset
A=&04 Get byte ignoring byte offset
A=&05 Read title,option
A=&06 Read current directory
A=&07 Read current library

A=&08 Read file names

ON EXIT,

A = 0 implies operation attempted

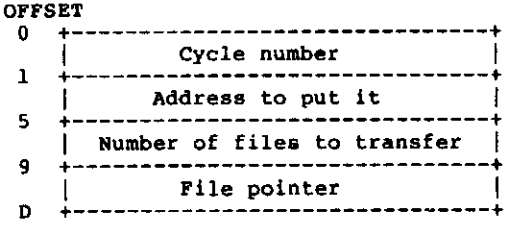
A=&01 to &04
The number of bytes is modified to show how much data has not been transferred (usually zero), the pointer value is updated (i.e. old pointer plus the amount transferred), and the offset is set to its current value. The offset is only defined after the call if you are not using the sequential file pointer.

A=&05
This returns in the area pointed at by the pointer the disc title of the currently selected disk and the users option. It is returned in this form:-
<title length><title><option>
The cycle number and option are single binary bytes.

A=&06
This returns the currently selected directory. It is returned in this form:-
<zero byte><length directory name><directory name><priv>
<priv> = &00 => Owner
<priv> = &FF => Public

A=&07
This returns the currently selected library. It is returned in this form:-
<zero byte><length library name><library name><priv>

A=&08
This returns the files in the current directory. The format of the control block is similar to that for sequential files:-



If the pointer is set to zero the search will begin with the first file. All are updated in a similar manner to the way pointers are updated for A=&01 to &04. The format of the filenames is as follows:
<length filename 1><filename 1><length filename 2><filename 2>.....
The number of entries which it is possible to read is limited by a 250-byte buffer. Attempts to move more than 250 bytes of data in one operation will produce the "No reply" error. The file pointer may be up to 255 only.

3.5 OSBGET

Call address &FFD7

Gets a byte from a specified open file.

ON ENTRY,
Y is the file handle

ON EXIT,
A contains the byte. If an attempt is made to read past the end of the file C will be set and A=&FE otherwise C is clear. X and Y are preserved.

3.6 OSBPUT

Call address &FFD4

Puts a byte to a specified open file.

ON ENTRY,
Y is the file handle and A contains the byte to put.

ON EXIT,
A, X and Y preserved.

4. The Printer Server and Printing.

4.1 The Printer Server

The Printer Server is a program available in ATOM BASIC for an ATOM or in EPROM for the BBC Microcomputer. When it is running, it will allow any station on the network to use a printer attatched to it. Once one station is using the printer, all other stations will not be able to contact the printer server until that printing job is finished.

Stations on the ECONET expect the printer server station to be number 235, although it is not strictly neccessary to use this number.

4.1.1 Using a BBC Microcomputer printer server

The printer server program for the BBC Microcomputer is held in a paged ROM. The computer which is to be used as the printer server must have both the Econet filing system ROM and the Printer Server ROM installed.

The BBC Machine printer server operates in background, which means that the machine can be used as a normal Econet station while the printer is running.

A separate manual is provided with the printer server EPROM and contains full operating instructions.

4.1.2 Using an ATOM printer server

The Printer Server program in ATOM BASIC can be run on any ATOM attached to the net. The Printer Server is located in directory $.PRINTER on the File Server Master Disc, and is file PSERV. It can be loaded into the relevant ATOM simply by logging on to the file server from that ATOM and using the BASIC LOAD command. After loading the program type RUN to start it.

4.2 Printing from a station

There are two steps to starting the Econet printer from a station. Both must be carried out on the BBC Microcomputer, on the ATOM/Systems start at step 2.

(1) Type

```
*FX5,4
```

to select the Econet printer.

(2) Type CTRL-B to make your machine attempt to contact the printer server. If contact is established your station number will appear on the printer, and everthing which appears on your VDU from now on, is sent to the printer.

If the printer server is busy, or not available for some reason the station will show the NOT LISTENING message, and BRK will be executed.

To finish sending output to the printer, type CTRL-C, which will tell the printer server that you have finished, and print several line-feeds to end your output. If the printer server loses contact with you during a printing session, the message TIME OUT will be printed on the printer, and the printer server will become free for use again. This will also happen if you don't print anything for a time (about 20 seconds) after you have typed CTRL-B.

4.3 Several Printer Servers

An ECONET station keeps the number of the printer server station in RAM. On the Atom and System computers at locations &022E/F (low byte in &022E) on the BBC Microcomputer it is stored in the operating system area. Whenever the machine is hard reset (CTRL BREAK or power-up on the BBC Microcomputer), this number is set to 235, the default printer server.

This number can be changed using the *PS command, which is loaded from the file server (see 2.9.6). e.g.

```
*PS 123
```

means that printed output will now be sent from your machine to station 123.

To change the printer server number from within an program, simply alter &022E/F (in your Atom or System computer) to be the new number, and type CTRL-B as before. On the BBC computer use OSWORD call with A=&13 (see section 11.8.1). It is therefore quite possible to have several printer servers which can be used by different stations on the network.

5 Talking to other user stations

5.1 Introduction

A number of commands are provided which allow for communication between similar stations. Some of these are loaded programs and some are in the Econet filing system ROM. It is possible to take a look at another station's screen (*VIEW), send a short message to another station (*NOTIFY) or completely take over another station (*REMOTE). The two stations in communication must be of the same type (either BBC Microcomputer or ATOM/System).

5.2 *VIEW

VIEW is a command program which is loaded from the appropriate library on the file server, (see section 2.6.1). In Atoms and Systems it is loaded at location &2800 and immediately executed. A station wishing to use VIEW must therefore be logged on to the file server and have the appropriate library selected.

The action of VIEW is to copy the screen of a distant user's station on to your screen. The two stations must be of the same type for this command to work properly.

If the distant machine is in a graphics mode, the users machine will also be placed in graphics mode (although this depends upon your graphics mode - see note below), so high-resolution pictures can also be viewed with this command. e.g.

```
*VIEW 100
```

will copy the screen of station 100 to our screen. You can also view a user by specifying his/her user identifier. For example:

```
*VIEW JOEY
```

will copy JOEY's screen to ours if JOEY is logged on to the file server. If JOEY is logged on at more than one station, then only one station at which he is logged on will be viewed.

Note that after the VIEW has been completed, your own cursor will appear on the screen and may spoil the image. This can be avoided by writing a BASIC program which does the VIEW and then does not return until ESCAPE is pressed:

```
10 *VIEW 100:REM  COPY THE SCREEN
20 REPEAT:UNTIL 0:REM LOOP FOREVER
30 END
```

This program is for the BBC Microcomputer, note that on the ATOM and SYSTEMS the keyword REPEAT should be replaed by DO and the delimiter ":" by ";". BASIC programs may also be written to do continuous VIEWs so you can keep constantly up to date on what is happening on another station.

NOTE
*VIEW will only work between similar stations. Attempting to *VIEW a

different type of machine will have unpredictable results.
On BBC machines you may get the error "Mode x", where "x" is the mode of the viewed station. This will occur if the machine which you are viewing is in a screen mode which uses more memory than the screen mode you are in. After an error of this kind, you can read the screen mode of the remote station using OSWORD call with A=&13 and the first byte of the control block set to &0A (see also section 6.1.2). The screen mode is returned in the second byte of the control block. This allows you to trap this error and automatically change mode.

5.3 *REMOTE

The REMOTE command is, like VIEW, a utility which is loaded from the file server.

The REMOTE command enables one machine to 'take over' another so that everything typed on the users machine is executed on the distant machine. Also, everything which is written to the distant screen is written to the users screen, and the distant keyboard is disabled.

The effect of this is that the user has control over a distant computer. For example, if a user on station 200 types:

```
*REMOTE 100
```

the user at station 100 will find that his keyboard does not respond, but everything typed at station 200 appears on his screen. The command may also be used with the user identifier, for example:

```
*REMOTE JEZZ
```

will have the same effect as specifying the station number provided JEZZ is logged on. As for *VIEW, if JEZZ is logged on at more than one station, only one station at which he is logged on will be taken over.

If user 200 now types RUN, he will run whatever program is in station 100. The program run will NOT be the one on station 200 because station 200 is no longer in touch with his own computer and is effectively acting as a terminal to station 100.

When the conversation between the machines is started the remote machine will have to stop what it is doing. On Systems and the BBC Microcomputer an error is generated. If a BBC Machine has Version 1 BASIC and is running a program with error trapping, the error generated will cause the error trapping loop to be executed. Version 2 BASIC on the BBC Microcomputer will not be able to trap the error generated and so the program will be stopped. On ATOMs the current program will be interrupted in all cases.

5.4 ESCAPE and BREAK in REMOTE

5.4.1 On BBC Microcomputers

Typing ESCAPE on the master machine (station 200 in the above example) during a REMOTE conversation will have the usual ESCAPE effects. The connection will not be severed, but program listings and programs out of control will be stopped. The ESCAPE key on the remote machine is disabled when the REMOTE connection is made.

Typing BREAK on the master machine will have the usual effect of BREAK. The connection is severed, and the remote machine is disabled until reset. Typing BREAK key on the remote machine resets the machine as usual but the connection is only severed until the master types something at his keyboard. As soon as this is done, the connection is re-made. Even if the remote machine is switched off and then on again, the connection will be re-made once the master starts to type something.

5.4.2 On ATOMs and Systems

Typing ESCAPE on the master machine (station 200 in the above example) during a REMOTE conversation will re-initialise the link between the two machines. This has a similar effect to typing ESCAPE in BASIC in that listings and programs which are out of control can be stopped without having to reset the machine by typing BREAK. A BRK instruction will again be executed, so an error message will occur when ESCAPE is typed.

The BREAK key cannot be disabled on the remote machine, and if the person who has been taken over hits break, his machine is reset and is no longer in contact with the 'master'. When this happens, the master machine may either be trying to send something to the remote machine, or waiting for an instruction from the remote machine. If it is trying to send, after a few seconds (typically 5), it will give up and the message NOT LISTENING will appear on the master screen. If it is waiting, it has no way of knowing that the remote machine is no longer interested and will continue to wait until it is reset (or remotely taken over by somebody else). This will have the effect of disabling the keyboard.

Similar effects will occur in the remote machine if BREAK is hit on the master machine.

5.5 *ROFF

*ROFF stands for 'Remote Off' and when typed on the master machine terminates the link between the two machines.

5.6 *NOTIFY

*NOTIFY is a command provided in the station EPROM for ATOMs and Systems but is a loaded program for the BBC Microcomputer which allows single line messages to be sent from station to station. The message can only be sent between similar stations.
e.g.

```
*NOTIFY 234 HOW ARE YOU BRIAN ?
```

sends the message HOW ARE YOU BRIAN ? to station 234. Just as with *VIEW and *REMOTE you can type something like:

```
*NOTIFY BRIAN HOW ARE YOU BRIAN?
```

On the BBC Microcomputer the message is sent directly and goes into the keyboard buffer of BRIANs machine. A beep occurs and the following

will be printed on the screen:

```
-- 100: HOW ARE YOU BRIAN? --
```

where 100 is the sending station number. No carriage return is printed so BRIAN can delete the message from the line before continuing to type in at his keyboard.

On ATOMs and Systems, the message is only received when BRIAN presses carriage return on his machine, so that the message does not suddenly break into a line that he is typing. Until then, the station sending the message is inactive. If you type ESCAPE on the sending machine, the message is not sent, and you are back in control of your machine.

When BRIAN types carriage return, the message:

```
100: HOW ARE YOU BRIAN ?
```

will appear, if station 100 sent the message. Nothing else in his machine will be affected, and after the message is printed, the line which he has typed will be executed as usual.

5.7 Protection

5.7.1 Protection from what?

It is possible to stop other machines on the network from using NOTIFY or REMOTE on you. The mechanics of this are described in more detail in section 10.6, and the specific machine implementations in sections 11.7 and 12.7, but for most users the commands below should be sufficient.

5.7.2 *PROT

This is a command program loaded from the File Server which makes your machine protected. Any machine which tries to REMOTE, VIEW or NOTIFY you will now not be able to contact you and will get a NOT LISTENING error. You are, however, still able to TRANSMIT and RECEIVE (see section 10.6), which means that you can still use file commands to contact the file server.

Station numbers 240-254 are privileged in that protection does not prevent them from using NOTIFY/REMOTE and VIEW on other stations. It is suggested therefore, that only one station on the network be given a privileged number, and its use be strictly controlled.

5.7.3 *UNPROT

*UNPROT is again a command program, which simply makes your machine unprotected.

6. Errors

6.1 Errors on the BBC Microcomputer

6.1.1 Error messages

If the file server objects to some command, a message will be displayed. Most of the common errors are self-explanatory, but some will produce a message:

F.S. Error xx

where xx is a hexadecimal number.

This means that the file server has sent an error number for which it doesn't have an associated string. Section 6.4 gives a full list of the error numbers and a brief explanation of each.

A 6502 BRK instruction is also executed, which will signal to BASIC that an error has taken place. BASIC will react as it normally does to errors. (See the BBC Microcomputer User Guide for details).

However, things are not quite as simple as they may seem. The BBC Microcomputer can only cope with error numbers in the range A8 to C0 from the Econet File Server, but the File Server can generate many more numbers than this range allows. To overcome this problem, error number A8 is really a composite number. If error number A8 is returned, then this means that an error number less than A8 is the true error number sent from the file server. This number can then be found by calling an OSWORD call. The string "F.S. Error xx" will display the real error number, but typing "PRINT ERR" (in BASIC) would return 168 (&A8).

6.1.2 OSWORD call to read error number

OSWORD call with A=&13.

The first byte of the control block (pointed to by XY - X lo-byte, Y hi-byte) should be set to &0A. The result returned in the second byte of the control block is an error number (see also section 11.8.1).

6.1.3 Line jammed - error number &A0

This error is returned if there is continuous information on the data lines. This may be due to a fault in another station or in one (or both) of the terminators.

6.1.4 Net error - error number &A1

This message is returned if an error occurs during transmission. It may be due to an unsuitable clock speed for the network, a fault in the terminators or a fault in the network cable.

6.1.5 No clock - error number &A3

This message is returned if the Econet interface cannot detect a valid clock signal. After checking that you are plugged in to the network, the problem will probably lie either in the clock generator or the clock lines of the cable.

6.1.6 Bad TXCB - error number &A4

This message indicates that a bad transmit block was used (see section 10.2).

6.2 Errors on the ATOM and Systems

6.2.1 Error Messages

The behaviour of ATOMs and Systems to errors is essentially the same as the BBC Microcomputer. Either an English error message or the message:

F.S. Error xx

will be displayed followed by a 6502 BRK instruction. However, the ATOM and Systems do not use error numbers less than A8 so error number A8 is not a composite number.

6.3 Errors common to all machines

6.3.1 NOT LISTENING - error number &A2

This message may occur when any NFS command is executed, and means either that the machine to which the command is directed is not attatched to the network, or is not interested. For example, trying to save a file on a network without a file server, or when the file server program is not running, will produce the NOT LISTENING message. Similarly, the message will appear on trying to VIEW a non-existent station.

The message will also occur when the distant machine is protected (see section 5.7) and REMOTE, VIEW or NOTIFY is attempted.

6.3.2 NO REPLY - error number &A5

This message is less likely to occur and generally means that some operation has failed in the middle. For example, if a user types LOAD "FILE" and at that moment, the disc is removed from the file server, or the file server is disabled in some way, it is likely that the NO REPLY message will occur.

6.3.3 Channel - error number &DE

This will occur if the Econet filing system memory is corrupted or if you switch your machine off and then on again while logged on. To rectify you will need to log on again.

6.4 List of Errors

This is a comprehensive list of errors, most of which will be turned into brief English strings before being sent to the client.

However, it is hoped that the list will be useful when writing programs using the file server interface (see chapter 13), and will provide a reference for the few errors which are returned to a user as hexadecimal numbers.

The word 'object' in the following descriptions refers to a file or a directory.

ERR (hex)	Description
13	USRMAN.RESTART called twice
14	Object not a directory
15	User not logged on
16	Machine zero is invalid
21	Cannot find password file
27	Password file syntax error
29	Object "$.PASSWORDS" has wrong type
31	STRMAN.RESTART called twice
32	SIN = 0
33	REF COUNT = &FF
34	REF COUNT = zero
35	File too big or zero length
36	Invalid window address
37	No free cache descriptors
38	Window ref count > 0
39	Big buffer already in use
3A	Invalid buffer address
3B	Ref. count = &FF
3C	Store deadlock
3D	Arith. overflow
42	Broken directory
46	Attempting to set WR attributes to a directory
48	Not defined
4C	No write access
4E	Too many entries asked for in EXAMINE
4F	Bad ARG to EXAMINE.
53	SIN not for start of chain
54	Disc not a file server disc
55	Both sector maps corrupt
56	Illegal drive number
57	Map seq. nos. differ by >1
58	Object size zero
59	New map doesn't fit in old space
5A	Disc of same name already in use
61	RNDMAN.RESTART called twice
64	All handles open - handle table full
65	Object not open
66	Cannot copy file handles
67	RANDTB full

69	Object not a file
6D	Invalid arg to RDSTAR
6F	GETBYTE; byte after last in file
71	Invalid number of sectors
72	Store address overflow
73	Accessing beyond file end
74	Invalid SIN
83	Too much data from client (SAVE)
84	Wait bombs out
85	Invalid function code
86	Undefined
8A	File too big
8C	Bad privilege letter
8D	Excess data in PUTBYTES
8E	Bad INFO argument
8F	Bad arg to RDAR
A0	Line jammed
A1	Net error
A2	Not listening
A3	No clock
A4	Bad TXCB
A5	No reply
AD	Mode x
AE	User not logged on
AF	Types don't match
B0	Renaming across two discs
B1	User id. already exists
B2	Password file full
B3	Maximum directory size reached
B4	Directory not empty
B5	Trying to load a directory
B6	Disc error on map read/write
B7	Attempt to point outside a file
B8	Too many users
B9	Bad password
BA	Insufficient privelege
BB	Incorrect password
BC	User not known
BD	Insufficient access
BE	Object not a directory
BF	Who are you?
C0	Too many open files
C1	File not open for update
C2	Already open
C3	Entry locked
C6	Disc full
C7	Unrecoverable disc error
C8	Disc number not found
C9	Disc protected
CC	Bad file name
CF	Invalid access string
D6	Not found
DE	Channel
DF	End of file

FD	Bad string (filename etc.)
FE	Bad command

7 Supervising the network

7.1 General principles

It is important that someone is appointed to supervise the ECONET network. Installing a network requires some advance planning, and running the ECONET efficiently requires that some routine tasks are performed regularly, particularly with reference to the file server. Starting the file server is one aspect of the supervisor's job described in this chapter. Another important task is 'ARCHIVING' which means taking regular copies of file server discs as protection against their loss or damage. This chapter is written for the network supervisor and may be ignored by most network users.

7.2 Hardware Requirements

The file server is written in 6502 machine code and is therefore restricted to running on 6502 machines. It is designed primarily for use on ACORN SYSTEM 3, 4 and 5 computers with one or more disc drives. A version will also be available for running on a BBC Microcomputer with second 6502 processor and disc drive(s). The BBC Microcomputer version will have the advantage of being faster than the Systems 3, 4 or 5.

The system running the file server should have at least 40k of RAM to support dual 40 track discs, and it is recommended that 48k of RAM be used with dual 80 track discs.

The amount of free memory in the machine will affect the performance of the file server with heavy loading (20+ users using the system constantly), and for systems which are going to be subjected to such a load, 40 or 48k of memory is recommended.

The File Server is loaded at &2400 on the Systems, and the 40-48K of memory should be available from there upwards. This will also take care of the requirements of the initialisation program written in a version of BASIC loaded at &A000. Page zero should also be available.

The maximum amount of disc space supported by this file server is 800k on two double sided 80 track 5.25 inch floppy discs.

7.2.1 A Useful File Server configuration

This section describes the cards which should be incorporated into either a System 3/4 or a System 5 to provide a useful file server.

System 3/4

CPU card
VDU card
Econet Interface card (preferably station 254)
Memory: Either 5 (or 6) 8K RAM cards linked at &2000, &4000, &6000, &8000 and &A000 (and &C000 if required).
Or One 32K Dynamic RAM card at &2000, &4000, &6000 and &8000 plus one (or two) 8K RAM cards at

&A000 (and &C000 if required).
Disc interface card and one or two disc drives
Econet software in EPROM at &E000
DOS software in EPROM at &F000

System 5

CPU card with combined DOS/Econet EPROM.
VDU card
Econet Interface card (preferably station 254)
Memory: Two 32K Dynamic RAM cards linked at &2000, &4000, &6000, &8000 and &A000, &C000.
Disc interface card and disc drives

7.3 Discs provided

7.3.1 File server and DFS discs not compatible

Because the File Server provides a completely different file system to the single user disc systems, discs which are used in the file server contain data structures which are not understood by the System DOS or the BBC Microcomputer DFS. The result is that DFS discs cannot be used in the File Server and File Server discs cannot be used under DFS.

Throughout this manual discs for use under the BBC Microcomputer DFS or System DOS will be called DFS discs, and those for use with the file server, file server discs.

The following two discs are provided with the ECONET system:

7.3.2 The NETWORK UTILITY DISC

This is a DFS disc and contains useful DFS utilities for copying discs, formatting discs etc. These are documented in the appropriate disc system manual. It also contains the following programs:

a) The File Server itself (FS)

b) The disc initialisation program and EXEC file (MPINIT) and INITxx (xx is the version number).

c) SBASIC - the BASIC used to run the initialisation program.

7.3.3 The FILE SERVER MASTER DISC

This is a file server disc which is initialised and contains REMOTE and VIEW for Systems, BBC Microcomputers and Atoms, various other utilities, and the printer server.

This is provided so that the file server can easily be started up without having to go through the process of initialising a file server disc when still unfamiliar with the file server itself. Initialising a file server disc and copying DFS discs to file server discs is

described in sections 7.5 and 7.9 respectively.

7.3.4 Disc care and handling

Discs should be handled with care to avoid physical damage or damage to the recorded information. The following guidelines should be observed:

Do not try and remove the circular magnetic disc from the square, black protective jacket covering it.

Do not touch the exposed recording surfaces.

Avoid dust. Put the discs back into their envelopes when they are not in the disc drive.

Do not bend them, drop them on the floor or put heavy objects on them.

Keep them in a storage box designed for the purpose.

Keep them away from strong magnetic fields such as those generated by televisions, monitors, tape recorders, telephones, transformers, calculators etc.

Avoid excessive heat, moisture and direct sunlight.

Only use felt-tipped pens to write on the labels and don't press hard.

Insert discs into the drive carefully. If it rotates noisily open the drive door and adjust it.

Information is packed quite densely onto the disc, so they are sensitive to even very small scratches and particles of food, dust and tobacco.

7.4 Starting the file server program

7.4.1 Basic steps

To run the file server program you will need an initialised file server disc. Section 7.5 deals with this initialisation, but you can start the file server with the FILE SERVER MASTER DISC provided. So to get going all you need do is follow the instructions below for a routine start of the file server program.

7.4.2 Routine starting procedure

This is the procedure to start the file server program whenever you have initialised file server discs ready. Errors occuring on the file server machine will be of the same form as error messages sent to a remote station - most will be in English, with less frequent ones in the form F.S. ERROR xx (see chapter 6).

BEGIN HERE

Press BREAK (followed by *DISC if necessary) to initialise the disc system on a BBC Microcomputer.

Press DELETE BREAK on an Acorn System computer (holding both keys down together).

Insert the NETWORK UTILITY DISC.

Type *FS 'RETURN' on the BBC Microcomputer

Type FS 'RETURN' on a System computer

It is very important that the memory is completely reliable so the file server will now start and test all the memory. While it is testing memory it will display:

ACORN File Server III.xx

Testing Memory

This will take several seconds. Just wait patiently for it to finish. If faulty memory is found the address where the fault occurs will be reported.

If everything is OK, the file server program asks some questions starting with:

1) DATE (DD/MM/YY) (xx/yy/zz)=

Insert a file server disc. (The FILE SERVER MASTER DISC if this is the first time you are starting the program).

Enter todays date in DD/MM/YY format. The file server will make a guess at the date (xx/yy/zz) and if this is correct, simply type carriage return to continue.
NOTE that the date will be stamped on all save operations taking place during the session, so it is worth making sure that the date entered is correct.

2) DRIVES=

Enter the number of drives on your system.

If only one drive is specified on a dual drive system, drive 0 will be assumed by the file server. It is not possible to use drive 1 by itself on a dual drive system.

3) COMMAND =

At this point, 5 options are possible:

S(tart) the file server
I(initialise) a new disc
C(onvert) a DFS disc

Q(uit) exit the file server.
A(gain) return to DATE prompt.

To continue, type S.

4) USERS =

Before you answer this prompt, make sure that the NET UTILITY DISC is removed, and that each drive contains a FILE SERVER FORMAT DISC.

Then simply enter the maximum number of users you expect to use the file server during this session.

The file server should now respond with:

STARTING - READY

at which point it can respond to requests from stations on the ECONET.

The File Server will not start if none of the discs in the system contain a password file, as it will then have no way of checking users who are trying to log on. The FILE SERVER MASTER DISC has a password file and a user called SYST with no password set. You can log on to the file server from a station using this identifier. Read the following sections of this chapter to find out how to proceed.

If you haven't already done so, it is probably a good idea to read through chapter 2 of the manual before continuing, to get a general view of the filing system. In particular, it will be helpful to understand the ideas of the ROOT and USERS ROOT directories (section 2.3.5).

7.4.3 File Server Monitor

It is possible for the file server to produce a display giving information on the commands received. The monitor is swtiched on or off by hitting the "m" key. When the monitor is on, every time a command is received by the file server, it displays the command and the station number of the station which sent the command.

7.5 Initialising file server discs

7.5.1 Why necessary?

In order to support many users on one disc, several data structures must be placed on a disc before it can be used in the file server.

NOTE - that the format of DFS discs is very different, and the file server will reject these if used.

The initialisation process is in three stages:

1/ Format the disc in DFS.

2/ Execute MPINIT to map the disc.

3/ Create a root directory.

These are discussed in the following sections.

7.5.2 Formatting the disc

This "soft-sectors" the disc using a disc system utility called FORMAT. 40-track discs should be formatted using the utility FORM40, 80-track discs using FORM80. The program operates on the currently selected drive, so use the *DRIVE command before executing the FORMAT (see DFS manual). (See also section 7.10).

7.5.3 Initialising the disc maps

With the NETWORK UTILITY disc inserted, type *EXEC MPINIT. The BASIC interpreter will be entered, and a program loaded and run. Several questions are displayed as follows:

```
DISC NAME:
```

and a name of up to 16 characters (spaces not allowed) should be entered. The file server will use this name to refer to the disc, and it should be chosen to indicate what is on the disc.

```
NUMBER OF TRACKS PER SIDE:
```

Simply enter the number of tracks on ONE side of the disc (typically 40 or 80).

```
DOUBLE SIDED?
```

Answer Y or N depending on whether the disc is single or double sided.

```
DRIVE:
```

Answer with the drive which holds the disc to be initialised, and make sure that the file server disc is inserted in that drive (the program will remind you).

There will be a short wait (approximately 30 seconds or so) while the program constructs the map. There will then be a disc transfer, and the message

```
DISC MAPS INITIALISED
```

indicates the end of the program.

7.5.4 Creating the ROOT and PASSWORD file

This final stage must be done in the file server itself, so start the file server up (using the NETWORK UTILITY DISC), entering "1" at the DRIVES prompt. At the COMMAND prompt, type I (for initialise), making sure that the disc to be initialised is in the machine in drive 0.

The prompt

```
PASSWORD FILE?
```

will appear. It is not neccessary for a disc to have a password file (see section 7.6), so type Y or N here.

If Y, you will be repeatedly prompted for USERID =, at which point you should enter the user identifiers of people you wish to log on with this disc. Their passwords are intialised "empty", and should be set using the *PASS command from a network station.

Typing carriage return at the prompt will finish the initialisation process and you will be returned to the COMMAND prompt, at which point you can re-execute any of the 5 options.

The disc is now ready for use.

7.6 Managing the password file

So that the file server can check users' passwords when they log on, there must be at least one disc in the system which contains a password file. If there is no password file, anybody trying to log on will get the error PASSWORD FILE NOT FOUND. It is therefore up to the operator of the file server to make sure that when people are trying to log on, there is a relevant password file available.

It is not neccessary to have a password file on every disc, and in fact not very desirable since it is then quite possible that one user will have several passwords. It is suggested that each group of people that uses the file server (school classes etc.) has ONE disc only with a password file, and if other discs are required for a group, that these should be initialised without a password file.

On a system with two drives, it is then possible to use the password file disc as a master disc, which is always in the system, with other supplementary discs being used on another drive.

On a single drive system, managing with a single password file is more difficult since the master disc may have to be removed from time to time. In this case, it is probably still preferable to have one password file only, rather than duplicating the file on each disc. It is then neccessary to make sure that everyone who should be logged on is logged on before removing the master disc.

If there are two discs with password files in the system at once, the file server will only use one of them, selected when the system is started up, and then re-selected each time a disc is changed. The file server selects the file by simply searching each disc in turn, starting at drive zero, until it finds a password file. If there is a password file on both drives, the one on drive zero will always be selected. Note that this search is always done whenever a disc is changed, so removing the disc on drive zero, and replacing it with a disc without a password file would result in the password file on drive one becoming selected.

7.6.1 Forgotten passwords

If somebody forgets a password, a system user has to delete the relevant user identifier, and then re-create it (see section 7.7.2). This has no effect on the user's files, but restores the password to

"". It can then be reset as usual.

7.7 Privileged users - setting privilege

A privileged (or "system") user is allowed three sets of facilities which are denied to other users:

1/ Ownership of the root directory on any disc.

2/ The ability to create and delete users

3/ The ability to grant privilege to other users.

A system user is therefore able to create and delete entries in any directory on the system, and in particular, can create entries in the root. This enables new user root directories to be created, and allows libraries to be maintained etc.

The *PRIV command can be used by a system user to make other users system or non-system users e.g.

```
*PRIV ALAN S
```

makes ALAN a system user

```
*PRIV ALAN
```

makes ALAN a non-system user again.

7.7.1 The user SYST

Since only system users can create system users, there must be one system user to begin with. Therefore, a system privileged user called SYST is always automatically created in a password file . This user should have a password set as soon as possible after initialising a disc, as anybody logging on as SYST has complete access to all discs on the system.

SYST can be used to make users privileged, and create useful entries in the root.

Apart from being a system privileged user, SYST has no other special abilities, and once other system users have been created, could be made into an unprivileged user or removed completely if required.

7.7.2 Creating and deleting users

The command *NEWUSER creates a new user in the password file:

```
*NEWUSER JOEY
```

creates user JOEY who can now log on.

In order to give a new user some file space of his own, it is also neccessary to create a directory in the root with the same name as his user identifier. This can be done separately using the *CDIR command.

Removing a user is done with the command *REMUSER. Again, this does not affect his file structure which remains on the disc and must be deleted using *DELETE.

All these operations can ONLY be done by a system privileged user.

7.7.3 Extending the Password file

This will have to be done if more system users are created. The File Server Master Disc has a utility, in the appropriate library, called "EXTENDPWF". To extend the password file, log on as a system privileged user (SYST) and type:

```
*EXTENDPWF
```

This will reserve space for another 14 users. Note that this routine uses memory locations &8C to &8F inclusive in BBC Machines.

It is important that you update the passwords file on the correct disc.

7.8 Changing Discs

In order to give a fast reponse to file commands, the file server keeps a lot of information about its discs in memory when it is running. This means that it is NOT POSSIBLE TO SIMPLY REMOVE AND CHANGE A DISC WITH THE FILE SERVER RUNNING.

If this is done, the file server will still be under the impression that it is dealing with the previous disc and WILL CORRUPT THE NEW ONE.

To change a disc with the file server running, first press the ESCAPE key, and keep it down until the file server responds.

If running on a dual drive system, the prompt:

```
DRIVE:
```

will be given. Answer with the drive which is to be changed.

On a single drive system, the file server will immediately go on to the next message:

```
CHANGING DRIVE - xx
```

where xx is the drive number.

Some disc transfers may now take place as the information in store is returned to the disc.

The message:

```
CHANGE DISC
```

will now appear, and the discs can be changed.

When this has been done, press the SPACE bar, and the file server will

restart with the new disc.

NOTE - it is not possible to run the file server with two discs of the same name, as people using the system would have no way of distinguishing between them.

7.9 Copying discs from DFS

This facility is provided so that files created under DFS and in particular using File Server I, can be copied to a file server format disc.

The destination disc should be an initialised file server disc with a root directory.

This program will work with a single drive system, but the discs will have to be repeatedly changed as the files are copied across. Note that in the following discussion the word "Qualifier" is used to refer to the DFS directories so as to distinguish them from file server directories.

1/ Start the file server, making sure that the file server disc is inserted. If you have a dual drive system, and have typed 2 to the DRIVES prompt, the File Server will expect a File Server disc to be on each drive at this point.

2/ At the COMMAND prompt, type C, for convert.

3/ The prompt

DRIVE:

will appear, followed by

SIDE:

Answer these with the drive and side of the DFS disc you wish to copy. Sides are numbered 1 and 2 for top and bottom surfaces. On a double-drive system, the DFS disc should always be on drive 1 (the right-hand drive usually).

4/ On a single drive system, the prompts INSERT DFS DISC, or INSERT F.S. DISC will appear. At these points, insert the relevant disc, and press the SPACE bar when done.

5/ The catalogue of the DFS disc will now be inspected, and a prompt

DIRECTORY FOR QUALIFIER "x":

will appear for each qualifier found. Give the directory name into which files under that qualifier are to be copied on the file server disc. If the directory does not exist, it will be created. If you do not want to copy files under some qualifier, type carriage return at this prompt, and those files will be ignored. If any error occurs, a message will be displayed, and you will be able to try entering the directory name again.

6/ Only 7 qualifiers can be dealt with at once, and if this number is reached, the program will start the copy immediately.

7/ Once all the qualifiers have been dealt with (or the maximum number reached) the files will be copied, one at a time, in blocks of &0F00 characters. This means that large files may require several changes of disc on a single drive system.

8/ If an error occurs during the transfer, the message CONTINUE? will appear. Type Y here if you wish to continue with the next file (the one with the error is not converted), or N if you wish to quit the program. It is possible that you will have files on a DFS disc which do not have legal file server names. In particular, file names with "." in them should be treated with care, as the file server will treat this as a directory name delimiter.

9/ The final prompt is AGAIN? which allows you to insert another DFS disc (NOT file server disc) and do another conversion. Typing Y restarts the program, N returns you to the COMMAND prompt.

7.10 ARCHIVING! - Data security

One of the most important duties of the network supervisor is to regularly make copies of the file server discs. This is because discs may become unreadable for a variety of reasons. Either rough handling or an operating mistake can destroy the information stored on disc. The following sections discuss a practical backup routine and what to do if a disc fails.

7.10.1 Backup routines

To protect the information saved onto the file server discs, the network supervisor should make regular copies of the discs. Where computers are used in business, industry or other activities which use large volumes of information, a standard routine has evolved. It is often called the "Grandfather, Father, Son" principle of copying information. Applying this principle to protecting the file server information will be beneficial. It works as follows:

```
Day 1  MASTER copied to SON
Day 2  MASTER copied to FATHER
Day 3  MASTER copied to GRANDFATHER
```

As you can see, it involves keeping three separate discs, each with a copy of the information from the master disc on it. On day 4 the master would be copied to the son again, and so the cycle continues.

The routines to copy discs which are available on the NETWORK UTILITIES DISC are described in the next sections.

7.10.2 Copying on the BBC Microcomputer file server

Before a disc can be copied to it must be formatted. To do this proceed as follows:

Insert the Network Utilities disc in drive 0 and type

*FORM40

or

*FORM80

You should use the former command if you have 40-track drives and the latter for 80-track drives.

Remove the utilities disc, put a new disc in its place. DO NOT FORGET TO CHANGE THE DISCS OVER. In reply to the question

Format which drive ?

reply with "0". And in reply to

Do you really want to format drive 0 ?

type "Y",

and wait while the computer formats the new disc. A series of numbers are displayed and when formatting is finished the word "Verified" appears. Now remove the new disc.

If you have a dual-drive,

Put the Network Utilities Disc in drive 0, put the new disc into drive 1 and type

*FSCOPY 01 <name of backup disc>

Where <name of backup disc> is the name you want the backup disc to have (you can use this to give the backup disc a different name to the master disc, thus distinguishing the two). Then replace the Network Utilities disc by the master disc and hit the SPACE bar, and wait while a copy of the master is made to the new disc.

If you have a single drive,

put the Network Utilities disc in drive 0 and type:

*FSCOPY 00 <name of backup disc>

and then follow the instructions on the screen to insert the master disc and the new disc alternately.

The utility "*FSCOPY" automatically copies both sides of double-sided discs which is essential for file server discs.

7.10.3 Copying on the system file server

Before copying to a new disc it must be formatted. To do this put the utilities disc in drive 0 and type

FORM40

or

FORM80

depending on whether you have 40 or 80 track drives.

Remove the utilities disc and replace it with the new disc to be formatted. DO NOT FORGET TO CHANGE THE DISCS OVER. In reply to the question

Do you really want to format drive 0

type "YES",

and wait while the disc is formatted. After a while a series of numbers will be displayed finishing up with the message "Verified". Now remove the new disc and put the utilities disc back in. Type

COPY 01

After the disc has stopped moving, remove it. Put the master disc in drive 0 and the new disc in drive 1. Press the SPACE bar and wait while a copy of the master is made to the new disc.

NOTE
It is essential that a complete copy of the master disc is made when backing up file server discs. Using routines to copy individual files (e.g. COPYF on the systems) will not produce a useful copy of a file server disc. If you are using double-sided discs you MUST copy both sides of the disc to produce a valid back-up copy. To do this you need to use COPY 01 and then COPY 23.

8. Installing a Network

8.1 Items required

To install an Econet network you will need the following items:

Two terminators with power supplies
One clock with power supply
Cable for the network
Sockets and plug connectors

Each of these items is discussed in the following sections.

8.1.1 Terminators

Two terminators are required, one for each end of the network cable. These are available ready-to-use housed in small plastic boxes. A socket on one end of the box is provided to connect the 8 volt power supply. On the other end of the box is a 5-pin DIN socket to connect to the end of the network cable.

8.1.2 Clock

One clock is needed for the network. It is important that only one clock is connected to the network. An Econet network clock is available housed in a box similar to the terminator boxes (don't get the two muddled up otherwise your network will not function correctly). The clock box has a power supply socket to accept an 8 volt DC supply and a 5-pin DIN socket to connect the box to the network. It is possible to alter the clock rate, this is dealt with in section 8.3.

8.1.3 Cable

The Econet network requires a four-core cable with earth screen. The cores should have a diameter of 0.9mm and the end-to end resistance of the network cable should be less than 15 ohms. The best cable to use is one which has its four cores arranged as two twisted pairs.

8.1.4 Sockets and plug connectors

Standard 180 degree 5-pin DIN sockets are used for connections to the network. Good quality DIN sockets should be used. Two 180 degree 5-pin DIN plugs should be joined by a piece of cable wired pin-to-pin. This is used to connect the stations to the network. The cable MUST be less than 2 metres long.

8.1.5 Suppliers

The various bits and pieces needed to make an Econet network can be obtained from the following suppliers. The list is by no means exhaustive and alternative parts may be suitable.

Terminators and Clock

These are provided by Acorn. The following parts numbers are

applicable:

Terminators Acorn part number AEH05
Clock Acorn part number AEH04
Power supply Acorn part number AAH08

Cable

BICC cable number CS7227 from
BICC in 2Km reels
Acorn, part number AEX03, by the metre.

For shorter networks (less than 300m) RS cable 367-921 in 100m reels from RS.

Sockets - 180 degree 5-pin

RS 478-273 (screw mounting) and RS 478-633 (circular mounting).

Lead connectors with 180 degree 5-pin plugs

Acorn part number AEX01 (1 metre of cable with two DIN plugs fitted and one T-piece).

Local hi-fi stores (make sure that the wiring is pin-to-pin)

Telephone numbers

BICC Helsby (09282) 2700
RS 01-253-3040
Acorn Cambridge (0223) 245200 - Sales Department.

8.2 Installing the network

Installing a network basically consists of laying the network cable, connecting DIN sockets where required and connecting the terminators and clock. However, before starting to set up your network, some precautions should be noted if it is to work reliably.

(1) Good soldering is essential. Dry joints or poor contacts which may cause intermittent operation are extremely difficult to track down and rectify. It is much better to spend some time (and money) in getting good connections in the first place.

(2) Short circuits will cause the system to malfunction. Hence you are strongly advised to use rubber insulating sleeves on the back of the DIN sockets to prevent possible short cicuits.

(3) Avoid cross wiring. The BICC cable recommended above has two white cores, one twisted around an orange core and one twisted around a blue core. It is very easy to get the white cores muddled up, if you do the system will not work. Unfortunately, the cores are not twisted very tightly and it will probably be necessary to strip off about 20cm of the outer insulation to determine which white is twisted around which colour. This is important to reduce cross-talk between the data and clock lines. If possible use a cable which has four differently coloured

cores. If you cannot do so, short circuit one twisted pair at one end of the cable and use an ohm-meter or battery and bulb to test for continuity. The polarity is also important. Thus the two data wires should not be switched around (similarly for the two clock wires).

The basic layout for the Econet network is shown in figure 1. The network consists of one cable without branches to which everything is connected by 5-pin DIN plugs and sockets. The sockets are all wired in parallel. The suggested colour codes for the connections are:

	RS	BICC	DIN pins
Data +	Black	White	1
Data -	Red	Orange	4
Earth	Shield	Shield	2
Clock +	White	White	3
Clock -	Black	Blue	5

The DIN plug pin number allocations are shown in figure 8.2

When installing the cable it is probably best to start at one end and work your way along, making sure that the similar coloured wires are not crossed before inserting each socket. For good network reliability, the sockets should be soldered into the network cable and the use of T-junction DIN sockets to connect stations avoided. This is essential for long or many-stationed networks since each T-junction introduces about 0.2 ohms into the network cable resistance.

A socket should be connected to each end of the cable to plug in a terminator. The socket for the clock is most usefully connected roughly half way along the cable. Other sockets are connected to the cable where a station is required. Note that any branches or spurs should be less than 2 metres long. Thus the wire connecting an Econet station to the network should be less than 2 metres long. The siting of the station sockets should be carefully planned so as to minimise spur lengths.

Figure 8.1 Econet network layout

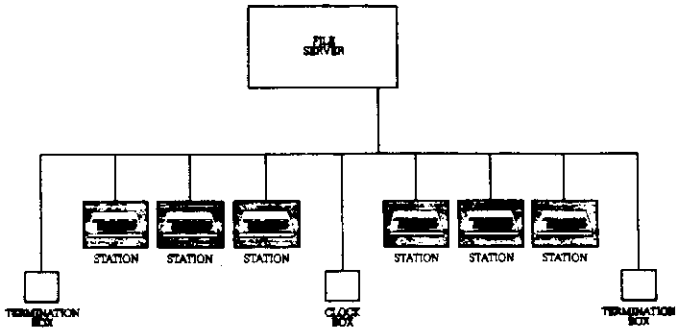
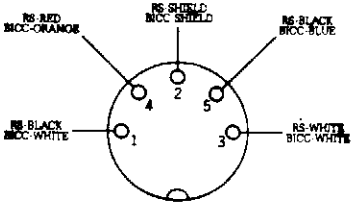


Figure 8.2 DIN pin number allocations



8.3 Setting up the network

By now you should be able to lay the main network cable and connect the DIN sockets. Before using the system you must do the following:

(1) Connect a terminator at each end of the cable, and connect the terminators to an 8 volt power supply. Connection of the supply to the terminators is indicated by a small red LED (light) near the power supply socket.

(2) Select a suitable clock rate. The clock rate you should select will depend upon distance from the clock to the furthest station. Using the clock rates available with the Acorn clock box, recommended maximum clock rates for different distances (in metres) are:

Distance	Maximum clock rate
100	307K
150	230K
240	153K
330	115K
500	76K

The following formula can be used to calculate the upper limit of the clock rate for a given clock to furthest station distance:

Rate = (velocity of signal in cable)/(distance * 4)

which for the BICC cable becomes:

Rate = 42 E6/(network length)

In practice a lower rate is often more reliable. The maximum clock rate is 307K for BBC Machines and System 5's and 200K for the ATOM and Systems 3 and 4. The minimum clock rate, imposed by the Econet hardware, is 70K.

To select a clock rate you will need to get inside the clock box. To do this remove the four screws, one in each corner of the base of the box. Carefully remove the top section of the box. In the middle of the circuit board in the box you should see two parallel rows of pins and a connector linking two of the pins. By the side of the rows of pins there is a series of numbers printed on the circuit board. These are the available clock rates. To select a given rate remove the connector from its present position and use it to connect the two pins to the right of the required rate. This is shown in figure 8.3, below, where a clock rate of 153KHz has been selected.

(3) Connect the clock to the network, as close as possible to the centre of the cable for maximum speed. Connect an 8 volt power supply to the clock. A small red LED near the power supply socket will light to indicate that the supply is connected.

(4) Each station connected to the network must have a unique identity code hardwired onto its Econet interface card. This is

achieved by bridging pairs of pins on two parallel rows of eight pins. The eight pins form a binary number, where a bridge is placed, the corresponding bit in the station number will be zero. This is illustrated in figure 8.4, where station number 15 has been set-up. The net will not work if two stations with the same station number are connected. Note that the following default station numbers are reserved:

254 = file server
235 = printer server

and station numbers 240 to 254 are privileged in that other stations cannot protect themselves from these stations (see section 5.7.2).

Figure 8.3 Selecting clock rate

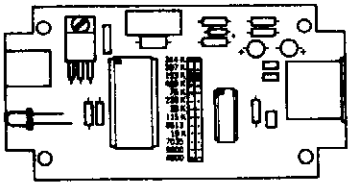
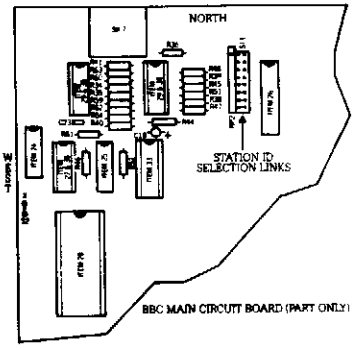


Figure 8.4 Selecting station identity number



8.4 Fault Diagnosis

If the network fails to operate when attempting to communicate with either the file server, the printer server or another station, then the following problems could be occuring:

"No Clock" message

This indicates that there is no clock signal reaching the Econet interface circuits. Check:

(1) Are you plugged into the network?
(2) Is power connected to the clock box? (Check the power indication LED).
(3) Is the clock box connected to the network?
(4) Are the clock lines open or shorted?

"Net Error" message

Check

(1) Is the clock speed suitable (see section 8.3)?
(2) Are the terminators connected to a power supply? (Check the LEDs)
(3) Are the terminators connected to the network?
(4) Is there network fault, e.g. short circuit, open circuit or crossed wires?

"Line jammed" message

This means that there is continuous information on the data lines. Do the following, testing to see if the fault is remedied after each stage.

(1) Reset all stations connected to the network.
(2) Remove all other stations from the network.
(3) Check for faulty terminators.
(4) Check for crossed wires in the network cable.
(5) Check for faulty local station.

"NOT LISTENING" message

Check:

(1) Are you sending your message to the correct station?
(2) Is the remote station plugged into the network, powered-up and in Econet mode?
(3) Is the network cable faulty?
(4) Is the local or remote station faulty?

"NO REPLY" message

Check:

Remote station has an attached and enabled peripheral, e.g. printer (is it switched on?), Disc drive etc.

9 Selecting a filing system

9.1 The BBC Microcomputer

The BBC Microcomputer can have several filing systems available other than the Econet system. The following commands are all used to exit from the current filing system into the one named.

*TAPE3	300 baud cassette
*TAPE12	1200 baud cassette
*TAPE	1200 baud cassette
*NET	Econet filing system
*TELESOFT	The telesoft mode of the prestel and teletext system
*TELETEXT	The terminal mode of the teletext system
*PRESTEL	The terminal mode of the Prestel system
*ROM	The cartridge ROM system
*DISC	The disc filing system
*DISK	Alternative spelling for above

Typing the command to enter the system you are already in has no effect. If you type the command to enter a filing system for which your computer is not equipped (i.e. you do not have the relevant filing system ROM) then the computer will respond with either

Bad command

or

No filing system

since it does not recognise the command.

9.2 Systems

To return to the disc operating system and not reset the computer type

*DOS

To return to the disc operating system and reset the computer press and hold DELETE and then press BREAK.

9.3 ATOMs

To return an ATOM to the cassette operating system type

*COS

to reset the computer and return to the cassette system press and hold SHIFT and press BREAK (while still holding SHIFT).

10 ECONET Primitives - Version 3

10.1 Introduction

The ECONET primitives are the basic operations which can be performed on the network to pass data and control between stations. These primitives will be implemented on the BBC Machine, and consequently, all address pointers are 32 bits long (to accomodate the address range of a 16 bit language processor). The bytes of these addresses will always be in 6502 order, i.e lsb first, and in this document will be labelled 0 to 3 for lsb to msb. The primitives are slightly different for the ATOM/Systems and these differences are shown in this chapter. This chapter only gives details of the primitives, the methods of calling the various functions are dealt with in chapters 11 and 12.

10.2 TRANSMIT

10.2.1 Purpose

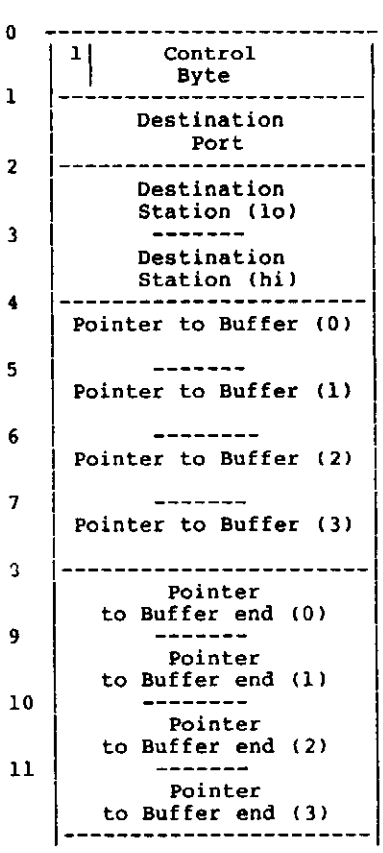
The TRANSMIT function sends a block of data from a buffer in one station to another station on the network. The receiving station must be expecting the message (see section 10.3) and the function will indicate if the message was successfully received.

10.2.2 The TRANSMIT Control Block

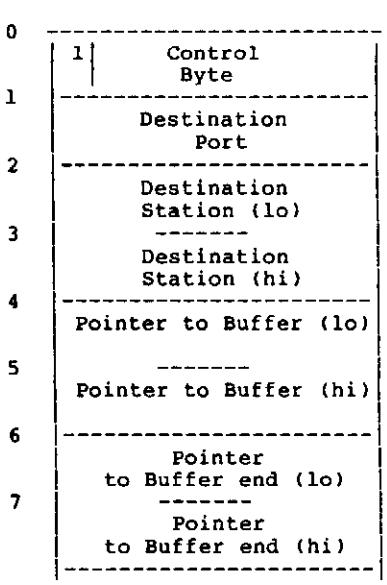
Arguments are passed to TRANSMIT in the form of a control block which can be set up anywhere in memory.

Transmit control block layout:

BBC Microcomputer



ATOMS and Systems



NOTE that buffer end = buffer + length, and that the buffer length must be positive, non-zero.

The STATION NUMBER is the station which is supposed to receive the message. Station numbers of zero and 255 are reserved as broadcast numbers.

The PORT NUMBER can be used to associate the data in the message with some task or process. The receiving station can be set up to listen for messages on a specific port or to any message addressed to it regardless of port number. If the receiving station is not listening to the port associated with the message, the message will not get

through.

Note that the port number is purely a software device and should not be confused with hardware I/O ports.

Ports &90 to &9F and &D0 and &D1 are used by the network filing system and should not be interferred with.

The CONTROL BYTE is a seven bit code which is sent to the receive block in the distant station.

10.2.3 Return from TRANSMIT

When transmission has completed, the control byte of the control block is altered.

Bit 7 = 0 => transmission has completed (BBC Machine only)
Bit 6 = 0 => Success
Bit 6 = 1 => Failure

Bit 5 will always be zero after a transmission.

On the BBC Machine, bits 0 to 4 will contain an error code if the transmission failed, or zero if the transmission was successful. In ATOM and SYSTEM implementations, these bits will be undefined after a transmission.

Low-level protocols have the characteristic that a failed transmit may, in fact, have been successfully received. Care should be taken that machines do not get out of step (see sections 13.9.8 and 13.9.9 for examples).

In the ATOM and SYSTEM, transmission will have completed when control returns from the transmit call, and bit 7 is therefore undefined after transmission. In the BBC Machine, transmission may continue asynchronously, and bit 7 must be polled to determine when a transmission has completed (see section 11.2).

10.3 RECEIVING Messages

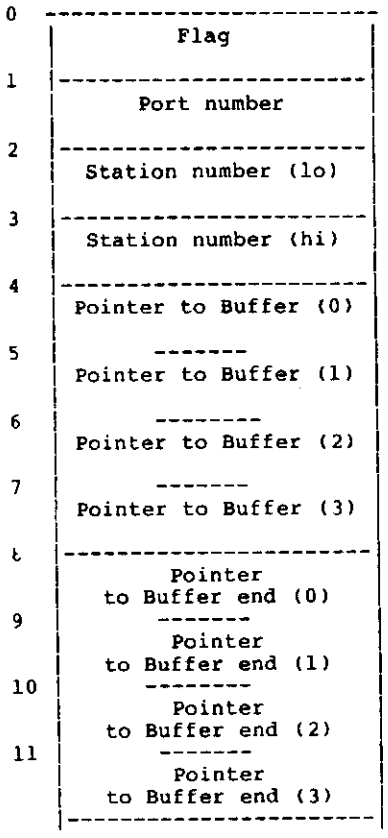
10.3.1 Introduction

Receiving into a non-BBC machine will be done by updating a data structure set up by the user. In the BBC machine, this data structure must be administered by the ECONET software itself, and accessed by operating system calls.

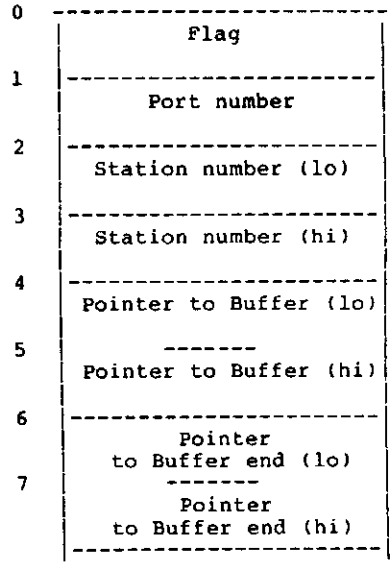
10.3.2 The RECEIVE CONTROL BLOCK

To receive data, a receive control block must be set up. Its format follows:

BBC Microcomputer



ATOM and Systems



The FLAG byte indicates the state of the RXCB. It should initially be set to &7F to indicate readiness to receive, will be altered on reception (see 10.3.4).

The STATION number indicates what station the control block will accept messages from. If this is 0, it will accept messages from any station (but only on the specified port).

The PORT number indicates on which port messages will be accepted. A port number of 0 indicates any port.

Both the station and port can be zero, and the station will then accept any message which is addressed to it from any station on any port.

10.3.3 The Receive Control Block After Reception

After a message is received, the following changes will occur to fields in the control block which accepted the message:

1) The top bit of the flag byte is set. The rest of the byte contains the control code sent from byte 0 of the corresponding transmit control block.

2) The station number, if originally zero, is modified to indicate which station actually sent the received message.

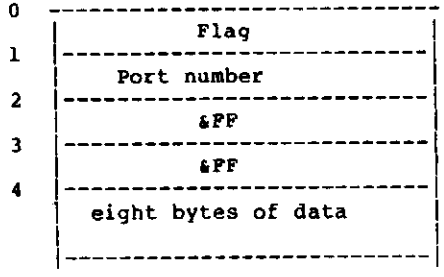
3) The port number, if originally zero, is modified to indicate the port of the received message.

4) The Buffer End pointer (both bytes) is modified to point to the end of the message in the buffer, if the message length is shorter than the buffer length. Note that messages longer than the buffer length will not be accepted.

The 'end of the message' is the byte after the last byte of the message.

10.4 Broadcast

If a station sets up a suitable control block, as given below, the data within the block can be received by ANY BBC Machine with a suitable receive block set up (the broadcast facility is not available for ATOM and SYSTEM computers). The receive block must be set up to listen to station &FF or station &00 and with the correct port (or port 0). The TRANSMIT control block for broadcast is:



10.5 Immediate Operations

10.5.1 Introduction

The following primitives differ from receive and transmit in that the station invoking the operation does not need the cooperation of the distant station.

10.5.2 PEEK

Takes a block of memory from the remote machine and puts it in a buffer in the local machine.

Control block:

BBC Microcomputer

0	Flag = &81
1	Zero
2	Station number (lo)
3	Station number (hi)
4	Pointer to BUFFER in local machine (0)
5	Pointer to BUFFER in local machine (1)
6	Pointer to BUFFER in local machine (2)
7	Pointer to BUFFER in local machine (3)
8	Pointer to Buffer end (0)
9	Pointer to Buffer end (1)
10	Pointer to Buffer end (2)
11	Pointer to Buffer end (3)
12	PEEK address in distant machine (0)
13	PEEK address in distant machine (1)
14	PEEK address in distant machine (2)
15	PEEK address in distant machine (3)

ATOMs and Systems

0	Flag = &81
1	Zero
2	Station number (lo)
3	Station number (hi)
4	Pointer to BUFFER in local machine (lo)
5	Pointer to BUFFER in local machine (hi)
6	Pointer to Buffer end (lo)
7	Pointer to Buffer end (hi)
8	PEEK address in distant machine (0)
9	PEEK address in distant machine (1)
10	PEEK address in distant machine (2)
11	PEEK address in distant machine (3)

10.5.3 POKE

POKE puts a block of memory from the local machine into an area in the remote machine.

The control block is the same as for PEEK, except that the operation flaq (byte 1) is &82 instead of &81. The effect is to move the BUFFER of the local machine into the address specified in the remote machine.

10.5.4 REMOTE SUBROUTINE JMP

This operation forces a subroutine jump to be made in the remote machine to an address specified by the local machine. A block of memory is passed to the remote machine before the call is made, so that arguments may be passed to the routine. The implementation of this call, and the method used to pass the arguments is machine dependant. If the receiving machine cannot buffer the arguments, it will not accept the call.

Notes:
This call does not work across the TUBE in BBC Microcomputers.
Interrupts will be disabled in ATOMs and BBC machines on entry to the call. In the BBC Microcomputer these should be enabled if the JSR routine is a long one (typically longer than one millisecond execution time).

Control block:

BBC Microcomputer

0	Flag = &83
1	Zero
2	Station number (lo)
3	Station number (hi)
4	Pointer to BUFFER in local machine (0)
5	Pointer to BUFFER in local machine (1)
6	Pointer to BUFFER in local machine (2)
7	Pointer to BUFFER in local machine (3)
8	Pointer to Buffer end (0)
9	Pointer to Buffer end (1)
10	Pointer to Buffer end (2)
11	Pointer to Buffer end (3)
12	Remote station Start Address (0)
13	Remote station Start Address (1)
14	Remote station Start Address (2)
15	Remote station Start Address (3)
16	

ATOM and Systems

0	Flag = &83
1	Zero
2	Station number (lo)
3	Station number (hi)
4	Pointer to BUFFER in local machine (lo)
5	Pointer to BUFFER in local machine (hi)
6	Pointer to Buffer end (lo)
7	Pointer to Buffer end (hi)
8	Remote station Start Address (0)
9	Remote station Start Address (1)
10	Remote station Start Address (2)
11	Remote station Start Address (3)
12	

The BUFFER in the local machine contains the argument block.

10.5.5 Remote Procedure Call

This primitive allows remote procedure calls to be made in a remote machine. A procedure is identified by a 16 bit number and is completely independant of program counter.

In the BBC Machine, the mechanism for reading the arguments to the call is the same as for REMOTE JSR.

An OPERATING SYSTEM PROCEDURE CALL primitive is provided with an identical control block to the procedure call. This primitive has a control flag of &84. This should not be used by ordinary users.

The control block follows:

BBC Microcomputer

0	Flag = &84
1	Zero
2	Station number (lo)
3	Station number (hi)
4	Pointer to BUFFER in local machine (0)
5	Pointer to BUFFER in local machine (1)
6	Pointer to BUFFER in local machine (2)
7	Pointer to BUFFER in local machine (3)
8	Pointer to Buffer end (0)
9	Pointer to Buffer end (1)
10	Pointer to Buffer end (2)
11	Pointer to Buffer end (3)
12	Procedure number (lo)
13	Procedure number (hi)
14	

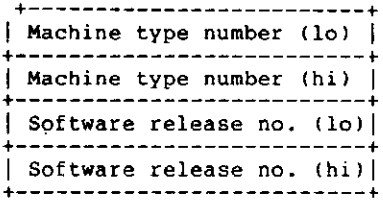
ATOMs and Systems

0	Flag = &84
1	Zero
2	Station number (lo)
3	Station number (hi)
4	Pointer to BUFFER in local machine (lo)
5	Pointer to BUFFER in local machine (hi)
6	Pointer to Buffer end (lo)
7	Pointer to Buffer end (hi)
8	Procedure number (lo)
9	Procedure number (hi)
10	

10.5.6 MACHINE TYPE

This primitive is designed to give some indication of the type of another machine on the network. It is effectively a peek of a store area which has been set by the low-level software in the distant machine.

In all ACORN machines, this area is set to be in the ECONET EPROM, and consists of the following:

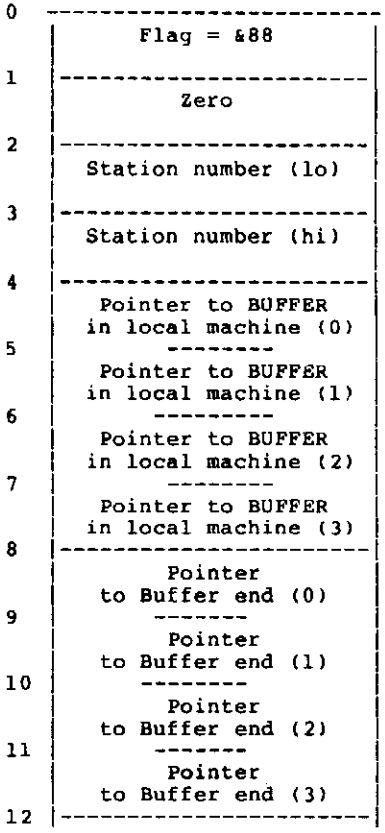


The machine type numbers and corresponding strings are:

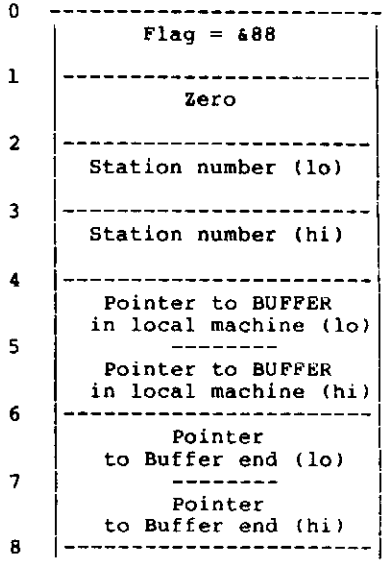
Type number	Machine
1	BBC Microcomputer
2	ATOM
3	System 3 or 4
4	System 5

Control block:

BBC Microcomputer

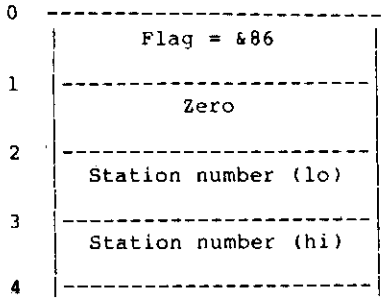


ATOM and Systems



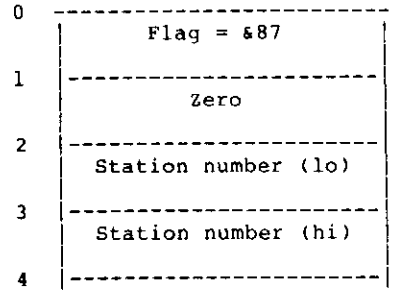
10.5.7 HALT

Halts the remote machine so that peeks and pokes can be made to the remote machine memory without interference from user processes. Interrupts are enabled in the halted state so that operating system functions may continue. In dual processor machines, only the I/O (host) processor can be halted. The control block is the same for all three computers:



10.5.8 CONTINUE

Continue user process after a HALT. In dual processor machines, only the I/O processor is affected. The control block is the same for all three machines:



10.6 Protection

It is possible to stop each of the ECONET primitives from having an effect on the local station. A PROTECTION MASK can be set, containing a bit for each of the immediate operations. The bits are assigned as follows:

```
Bit 0 = 1 => PEEK not allowed
Bit 1 = 1 => POKE not allowed
Bit 2 = 1 => REMOTE JSR not allowed
Bit 3 = 1 => User Procedure call not allowed
Bit 4 = 1 => OS Procedure call not allowed
Bit 5 = 1 => STOP not allowed
```

It is not possible to protect against CONTINUE or Machine PEEK. Stations with numbers 240 to 256 are privilaeged and cannot be protected against.

11 The BBC Microcomputer operating system interface

11.1 Introduction

This chapter describes the BBC Microcomputer machine operating system calls necessary to implement the Econet primitives described in the previous chapter. It should be noted that it is not necessary to select the network system as the current filing system to use most of the primitive network facilities.

11.2 TRANSMIT

11.2.1 Calling TRANSMIT

This is done by an OSWORD call:

OSWORD call with A=&10

On entry, X and Y point to the control block (X lo-byte, Y hi-byte).

If the transmit is initiated successfully, the control byte is unchanged. Otherwise, the control byte is set to zero. This will only happen if transmit is called while a previous transmit is still continuing under interrupts.

11.2.2 Polling the TRANSMIT block

This is done be an OSBYTE call.

OSBYTE call with A=&32

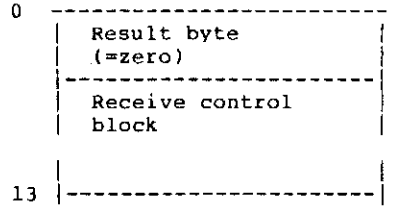
On exit, X contains the control byte of the specified block, or zero if the read attempt failed.

11.3 RECEIVE

11.3.1 Opening a Control Block

This is done by OSWORD call with A=&11

On entry, X and Y point to the control block (X lo-byte, Y hi-byte), preceded by a result byte which must be zero to indicate an OPEN:



On exit, the result byte contains a control block number.

If the control block has been successfully set up, the result byte will be non-zero and will contain the CONTROL BLOCK NUMBER which can be used to poll and read the control block.

If the attempt fails because the operating system has no more space for RXCB's, the result byte will be zero. There is sufficient space for 16 receive control blocks.

If this call succeeds, the receive block is OPEN for reception and may receive messages from stations on the network.

11.3.2 Polling a Receive block

This is done using OSBYTE call with A=&33

On entry, X is the control block number.

On exit, X contains the control byte of the specified block, or zero if the read attempt failed.

11.3.3 Reading a Receive block

This is done by OSWORD call with A=&11

On entry, X and Y point to a 13 byte OSWORD control block, the first byte of which is the receive block number of the block to be read (similar to creating a receive block - see section 11.3.1). Note that this byte should be non-zero to indicate a READ and not a SET.

If the receive block number is valid, the corresponding control block is read into a receive area. The control block is DELETED when read, so that subsequent requests to open a control block may result in the initial number being re-allocated, and the initial control block being overwritten.

If the receive block number is not valid, the receive area is not altered, and byte XY+0 is set to zero.

11.3.4 Deleting a Receive Block

This is acheived by OSBYTE call with A=&34

On entry, X holds the control block number.

NOTE: Deleting a control block before reading it may lose information received from a distant station.

11.4 Immediate Operations

All immediate operations are called using the same calls as for TRANSMIT.

11.5 Remote Subroutine Jump

Arguments in the BBC machine cannot be passed in registers since it is not possible to know the type of machine being called. The registers on calling are therefore undefined.

Reading the arguments is acheived by OSWORD call with A=&12:

On exit, bytes XY+0 and XY+1 contain the station number of the calling station and bytes XY+2 onwards contain the arguments passed.

On reception of a JSR/Procedure or OS Procedure call, the JSR and remote procedure protection bits are set by the low-level software. These will be reset to their previous state when the buffer is read. Care should thus be taken when using the buffer and it is recommended that only one JSR/Procedure or OS procedure call is enabled at once. Note that this will not protect against privileged machines (240-256).

In addition, an OSWORD call is provided to read the length of the buffer, and the size of its contents (OSWORD call with A=&13 and Reason code 9 - see section 11.8.1).

11.6 Remote Procedure call

When a remote procedure is called an EVENT will occur. When this occurs;

A will hold the event number (8)
X and Y will hold the procedure number.

Events can be disabled/enabled using OSBYTE call &0D and &0E with the event number in X (8 in this case). (See the BBC Microcomputer User Guide P.425).

11.6.1 Operating System Procedure call

In the BBC Microcomputer implementation, control is passed to the high-level code in the operating system and never reaches the user.

11.7 Protection

OSWORD call with A=&13 is used to read/write the protection mask (see section 11.8.1).

11.8 Other OSWORD calls

11.8.1 Reading and setting state information

OSWORD call with A=&13

On entry, XY point to a control block, the first byte of which is a REASON CODE. Depending upon the reason code, the following bytes either contain information to be set, or are reserved for results.

On exit, XY+0 is preserved. XY+n may be modified if the reason code

indicates a read operation.

The reason codes are as follows:

```
 0 - Read file server number (2 bytes)
 1 - Set file server number (2 bytes)
 2 - Read printer server number (2 bytes)
 3 - Set printer server number (2 bytes)
 4 - Read protection mask (1 byte)
 5 - Set protection mask (1 byte)
 6 - Read context handles in order URD,CSD,LIB (3 bytes)
 7 - Set context handles (3 bytes)
 8 - Read local machine station number
 9 - Read number of routine arguments and buffer size resp. (2
bytes)
10 - Read error number
```

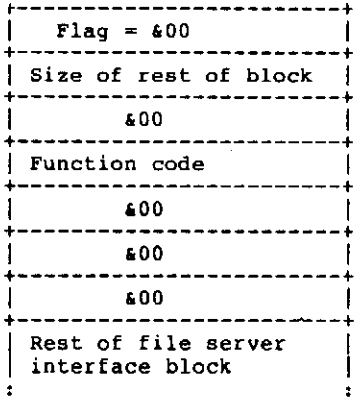
Reading and setting parameters connected with the file server (reason codes 0,1,6 and 7) should only be done when the network is selected as the current filing system.

11.8.2 File server/remote machine call

OSWORD call with A=&14

On entry,
XY+0 contains a flag which determines the action to be taken and the required contents of the rest of the control block:

Flag = 0 - Send rest of block to file server. This can be used with the file server interface described in chapter 13. The control block will be:

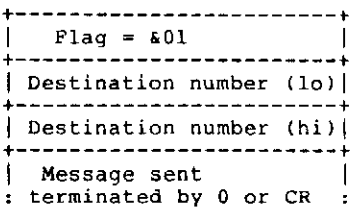


The "size" byte should indicate the number of bytes in the control block starting at the first byte after the "size" byte. The filing system will fill in the rest of the standard TX header (the zeros above) and send the request to the current file server. Changing the selected file server and the file server environment handles must be done with the appropriate OSWORD calls.

The reply from the file server will be in locations XY+2 onwards.

This will only work if the network is selected as the current filing system.

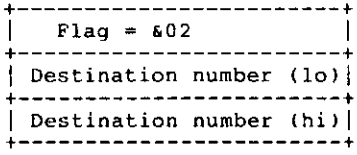
Flag = 1 - Send a text string to another station. The text string is put directly into the destination station's keyboard buffer. The control block is:



This will only work between BBC Machines.

Note
For both flags 0 and 1, the maximum length of the control block is 128 bytes if used via the TUBE (i.e. with a second processor). This puts an upper limit of 125 bytes on the message length. If used without the TUBE, the maximum lenght of the control block is 250 bytes.

Flag = 2 - Cause a "fatal error" in the remote machine. The control block will be:



After this call has been made, the call can be repeated with flag = 1 to send lines of text to the destination machine.

This OSWORD routine may not be called from within an interrupt routine.

11.9 Other OSBYTE calls

11.9.1 Switching a REMOTE off from within REMOTE

OSBYTE call with A=&35

No arguments, used to sever the connection between a computer and terminal in a remote conversation.

12 The ATOM and System operating system interface

12.1 Introduction

This chapter describes the ATOM and System calls necessary to implement the Econet primitives described in chapter 10.

12.2 Calling TRANSMIT

This is done by a JSR to address &023D with X and Y pointing to the control block (X lo-byte, Y hi-byte). The control byte must then be read explicitly to determine whether or not the transmit was successful.

12.3 Receiving

Receiving a block of data from the network is not done by a routine call, but by setting up a data structure indicating a willingness to receive from certain stations on certain ports. When a message comes in which matches the receive conditions, the data structure is updated to indicate that a reception of data has taken place.

The data structure controlling the acceptance of messages is the RECEIVE CONTROL BLOCK VECTOR which consists of zero or more RECEIVE CONTROL BLOCKS terminated by a zero byte.

In order to indicate that the station is ready to receive, two bytes at location &0230 must be set to point to the receive control block vector. In addition, bit7 of the FLAGS byte (location &023A) must be set to indicate the a receive control block vector exists. If this is not set, no messages will be received. Care must be taken when setting bit7 of the FLAGS byte that the rest of the byte is not disturbed, as the remaining bits are reserved for use by the low-level software.

The receive procedure is summarised below:

(1) Set up a receive control block anywhere in memory which has the flag byte set to &7F. It should point to a buffer which is big enough for the largest possible expected message.

(2) Set the receive control block vector pointers to point to the vector. This should be done with bit7 of &023A unset (zero) to avoid confusion if a message comes in while the receive control block vector pointers are being changed. NOTE that the vector must be terminated by 0 even it consists of only one receive control block.

(3) Set bit7 of &023A to 1, taking care not to disturb the rest of that location.

(4) When the flag of the receive control block has had the top bit set, the message data is in the buffer pointed to by the receive control block. A 7 bit control code from the transmitting station occupies the rest of the byte.

(5) Tidy up by setting the vector pointers to zero, when all possible messages have been received. This should again be done

with bit7 of &023A unset.

12.4 Immediate commands

All immediate operations are called using the same calls as for TRANSMIT.

Whenever a JSR or a Procedure call is received the JSR and Procedure protection bits are set. The user must then reset them appropriately after reading the arguments.

12.5 Remote Subroutine Jump

The processor registers pass arguments as follows. A will contain the first byte of the specified arguments, X and Y will contain the calling station number (X lo-byte, Y hi-byte). The remaining parameters will be put in an 8 byte buffer at &00C2 in ATOMs and &00A2 in Systems. If more than 8 bytes are sent. they won't be received.

12.6 Remote Procedure call

In ATOMs and Systems, the call will be to a fixed address in ROM which will indirect through location &238. (If you wish to change the indirection value you should do so with interrupts disabled so that the Econet doesn't interrupt while you are in the process of changing the value). The information passed in the call will be:

The bottom 8 bits of the procedure number in A.
The station number of the calling station in X and Y.
The remaining arguments will be put in the Procedure/JSR buffer.

12.6.1 Operating system procedure call

In the ATOM/Systerm implementation, control is passed to the high-level code in the operating system and never reaches the user.

12.7 Protection

In ATOMs and Systems, the protection mask is held at a known location (&23B) and can be directly accessed by the user.

12.8 Memory Allocation

12.8.1 Page Zero

ATOM

The low-level software on an ATOM uses locations &B0 to &CF inclusive. At least some of this will be corrupted when any network traffic takes place, even if not sent to your station.

The high-level software uses &D0 to &DD which will not be used unless

a high-level command or command program is executed.

REMOTE uses &ED to &F7 to ensure that other network commands can take place during a remote.

System

Low-level uses locations &90 to &AF

High-level uses locations &DD to &E9

Remote uses &80 to &8B

12.8.2 Page Two

These locations are the same for both ATOM and Systems.

Low-level:

&230 RXCBV lo
&231 RXCBV hi
&232 Interrupt address for receive (lo)
&233 Interrupt address for receive (hi)
&234-&237 Internal use
&238-&239 Procedure jump indirection
&23A FLAGS - bit 7 -> RXCBV exists
&23B Protection mask
&23C Internal use
&23D JSR instruction
&23E Address of TRANSMIT routine (lo)
&23F Address of TRANSMIT routine (hi)

High-level

&224 Handle for URD (file server)
&225 Handle for CSD (file server)
&226 Handle for library (file server)
&227 Random access sequence numbers
&228 Notifying Station (lo)
&229 Notifying Station (hi)
&22A Address for network errors (lo)
&22B Address for network errors (hi)
&22C Station to send file server commands (lo)
&22D Station to send file server commands (hi)
&22E Printer server station (lo)
&22F Printer server station (hi)

12.8.3 Other locations

The high-level software in an ATOM makes extensive use of the CLI buffer (&0100) during a file server command. This buffer is filled with carriage-returns (apart from the first 16 bytes) on exit from a file server command.

On a System 3/4, the catalogue buffer (&2000 to &2200) is used for receive and transmit buffers, and will be corrupted over a file server command.

13 File Server Interface

13.1 Introduction

The file server accepts requests from stations on the network (called CLIENTS) on a known port (the COMMAND PORT). This chapter will describe the type of requests which the file server is expecting, and the replies which it will give.

In general, the client will send a message to the file server which describes the operation which the file server is to perform. The message will also contain a REPLY PORT, and the client will then wait for a reply from the file server on that port. This reply will indicate whether any further messages should be exchanged in order to complete the operation.

13.2 The User Environment - Handles

As described in the main part of this manual, it is possible for a user to select a number of parameters which define how the file titles and commands which he sends to the file server are to be interpreted. There are, in fact, three factors which define the USER ENVIRONMENT:

1/ The Currently Selected Directory (CSD) - from which file titles which do not start with the root are resolved.

2/ The User Root Directory (URD) - which the user is returned to if he types *DIR.

3/ The Currently Selected Library (LIB) - where unrecognised commands will be looked up if not found in the CSD.

This environment is represented by three HANDLES which are created by the file server. Thus a command may be executed in any environment defined by the handles sent in the command request. It is therefore possible for a user to create several handles for several directories (using the OPEN command) and execute commands in different environments.

The handles are created when a user logs on, and deleted (CLOSED) when he logs off. Changing a disc is similar to logging on, in that the three previous handles are deleted, and three more created which refer to directories on the new disc.

In the simple case of just using commands, the three handles are managed by the client machine software, and the user is unaware of the way his environment is defined.

13.3 Definitions

Most initial messages to the file server start with five bytes in the following format, known as the STANDARD TX HEADER.

1 - Reply Port
2 - Function Code
3 - Handle for Users Root Directory

4 - Handle for Currently Selected Dir.
5 - Handle for a library directory.

The REPLY PORT has been described.

The FUNCTION CODE is a number telling the file server what operation is to be performed.

The three handles define the user environment for this command, as discussed above.

The STANDARD RX HEADER is a message format sent by the file server to the client:

1 - Command Code
2 - Return Code

The COMMAND CODE is 0 if the operation is complete. If not, the client has some more work to do, which may involve sending further messages to the file server.

The RETURN CODE is 0 if no errors occured during execution of the previous command step. If it is not zero, the return code is one of the error numbers listed in Appendix C and the rest of the message (bytes 3 onwards) is a string terminated by a carriage return (CR) which describes the error.

The COMMAND PORT is port number &99.

13.4 Command Line Decoding

Many operations of the file server are defined by a command line, and function code zero stands for the operation of decoding such a command line. When the file server receives a message with this function code, it will decode the following string as one of the commands defined in Appendix B. If no more information is required from the client, it will then immediately perform the operation and send a return code, and possibly some result of the operation.

If more information is required, a reply will be sent to the client indicating which command was found in the command line (the COMMAND CODE), and possibly the command parameters (file titles etc.)

The client then continues the operation, exchanging more messages with the file server.

The general format for the initial exchange of messages is:

Client -> File Server (command port)

Bytes 1-5 = standard TX header
" 6+ = command line, terminated with CR.

File Server -> Client (reply port)

```
Bytes 1-2  = standard RX header
  "   3+   = depends on command
```

13.5 Simple Commands

These commands require no further information from the client after the initial exchange of messages, the operation being executed by the file server on receipt of a command line.

```
*ACCESS
*BYE
*CDIR
*DELETE
*NEWUSER
*PASS
*PRIV
*REMUSER
```

13.6 SAVE - command code 1

Requests to save a file can be generated either by a command, or from within a program. The SAVE interface can therefore be seen in two sections, one dealing with the command line, the second dealing with a request from within a program.

1/ Client -> File Server (command port)

```
Bytes 1-5  = standard TX header
  "   6+   = command line
```

2/ File Server -> Client (reply port)

```
Bytes 1-2  = standard RX header
  "   3-6  = load address of file
  "   7-10 = execute address of file
  "  11-13 = size of file
  "   14+  = file title terminated CR
```

A save request from within a program starts at step 3.

3/ Client -> File Server (command port)

```
Bytes 1-5  = standard TX header BUT with acknowledge port in byte
             normally reserved for URD handle.
  "   6-9  = load address of file.
  "  10-13 = execute address of file
  "  14-16 = size of file
  "   17+  = file title terminated CR
```

4/ File Server -> Client (reply port)

```
Bytes 1-2  = standard RX header
Byte  3    = data port
Butes 4-5  = Block size
```

5/ Client -> File Server (data port)

Block of data, "block size" long. If the file size is zero, go to step 7.

6/ File Server -> Client (acknowledge port)

Single byte, value undefined.

Client continues to send data and receive acknowledges until all data has been sent. Step 6 is missed out after the last block has been sent, and the final acknowledge is as follows:

7/ File Server -> Client (reply port)

Bytes 1-2 = standard RX header
Byte 3 = Access byte (bottom 5 bits in order LWRWR)
Bytes 4-5 = Date (format as in section 3.2)

NOTE - if an error occurs at the file server end during the transfer, it will continue to receive blocks, but do nothing with them, and the nature of the error will be indicated in the final reply.

13.7 LOAD - command code 2

As with SAVE, LOAD can have two stages of operation. Firstly the decoding of a LOAD command line, and secondly the LOAD operation itself, which may be generated from within a program.

1/ Client -> File Server (command port)

Bytes 1-2 = standard TX header
" 6+ = command line

2/ File Server -> Client (reply port)

Bytes 1-2 = standard RX header
" 3-6 = load address found in command line.
Byte 7 = FLAG (&FF or &00)
Bytes 8+ = file title, terminated CR

The FLAG indicates whether a load address was found in the command line. If so, it is &FF, if not &00. If the flag is 0, the load address is undefined.

The LOAD operation starts at step 3.

3/ Client -> File Server (command port)

Bytes 1-5 = Standard TX header BUT data port in URD slot
" 6+ = file title, terminated CR

4/ File Server -> Client (reply port)

Bytes 1-2 = Standard RX header
" 3-6 = File load address

```
"   7-10 = File exectute address
"  11-13 = File size
"  14    = Access (as for SAVE)
"  15-16 = Date (as for SAVE)
```

5/ File Server -> Client (data port)

Data blocks, size undefined, until all data has been received (determined from file size). If file size is zero, go to step 6.

6/ File Server -> Client (reply port)

```
Bytes 1-2  = Standard RX header
```

As for SAVE, if an error occurs during the data transfer, "dummy" blocks will continue to be sent, and the error will be indicated in the final reply message.

13.8 Other Command Line Operations

All the following commands start by sending the command line to the file server on the command port. The reply, however, contains information which may be used at the client end. These replies are described below:

***INFO - command code 4**

```
Bytes 1-2  = Standard RX header
  "   3+   = Character string of information terminated by &80.
```

The information is formatted for use on Acorn machines.

***DIR - command code 7**

```
Bytes 1-2  = Standard RX header
Byte  3    = Handle for new CSD
```

***SDISC - command code 6**

```
Bytes 1-2  = Standard RX header
Byte  3    = Handle for new URD
  "   4    = Handle for new CSD
  "   5    = Handle for new LIB
```

***I AM - command code 5**

```
Bytes 1-2  = Standard RX header
Byte  3    = Handle for new URD
  "   4    = Handle for new CSD
  "   5    = Handle for new LIB
  "   6    = Boot option (lo 4 bits relevant)
```

***LIB - command code 9**

```
Bytes 1-2  = Standard RX header
Byte  3    = Handle for new LIB
```

UNRECOGNISED COMMAND - command code 8

```
Bytes 1-2  = Standard RX header
  "   3+ . = Command string, terminated CR.
```

13.9 Other Functions

These functions are designed for use from programs rather than typed commands and are identified by a function code other than zero.

Function codes

Code	Function
0	Command line decoding (see sections 13.4 to 13.8)
1	Save
2	Load
3	Examine
4	Catalogue header
5	Load as command
6	Find (OPEN)
7	Shut (CLOSE)
8	Get byte
9	Put byte
10	Get bytes
11	Put bytes
12	Read random access info.
13	Set random access info.
14	Read disc info.
15	Read logged on users
16	Read date/time
17	Read "end-of-file" status
18	Read object info.
19	Set object info.
20	Delete object
21	Read user environment
22	Set user option bits
23	log off
24	Read user info.
25	Read file server version number

These commands are discussed in the following sections.

13.9.1 Save - code 1

This is dealt with in section 13.6.

13.9.2 Load - code 2

This is dealt with in section 13.7

13.9.3 Examine - code 3

1/ Client -> File Server (command port)

```
Bytes 1-5  = Standard TX header
Byte  6    = ARG to examine
 "    7    = entry point to directory
 "    8    = number of entries
 "    9+   = name of directory to be examined
```

ARG defines the type of information to be returned by the file server.

```
ARG = 0 => all information, m/c readable
ARG = 1 => all information, character string
ARG = 2 => file title only
ARG = 3 => access + file title, character string.
```

ENTRY POINT - entry number in directory from which examine will take place.

NUMBER OF ENTRIES - how many entries are to be examined.

2/ File Server -> Client

```
Bytes 1-2  = Standard RX header
Byte  3    = number of entries examined
Bytes 4+   = depend on ARG.
```

A number of entries will be sent (byte 3). The machine readable forms are a defined number of bytes, and therefore no delimiters occur between them.

To allow easy display of the character forms, entries are delimited by a zero byte, and may include carriage returns.

13.9.3.1 ARG = 0

```
Bytes 4-13 = File title, padded with spaces
  "  14-17 = Load address
  "  18-21 = Execute address
Byte 22    = Access LWR/WR - bottom 5 bits
Bytes23-24 = Date

             23 - day
             24 - year since 81 (4bits)/month (4bits)

  "  25-27 = System Internal Name
  "  28-30 = Size of file
```

13.9.3.2 ARG = 1

Bytes 4+ = character string of all information.

Each entry is delimited by a zero byte, and the end of all information is indicated by a &80 byte after the final zero.

13.9.3.3 ARG = 2

Byte 4 = 10 (use by BBC Micro. OS)
Bytes 5-14 = File title padded with spaces

13.9.3.4 ARG = 3

Bytes 4+ = File title followed by formatted access string, Entry delimiters and final delimiter as with ARG = 1.

13.9.4 Catalogue Header - code 4

1/ Client -> File Server (command port)

Bytes 1-5 = Standard TX header
" 6+ = File title (possibly just CR)

2/ File Server -> Client (reply port)

Bytes 1-2 = Standard RX header
" 3+ = Last component of file title
Ownership of directory character
Current Disc name

Formatted as a character string

13.9.5 Load as command - code 5

1/ Client -> File server (command port)

Bytes 1-5 = Standard TX header
" 6+ = File name

Remaining steps exactly as for LOAD, except that the file title is looked up in several directories (as described in section 2.6), rather than just the current one.

13.9.6 Find(OPEN) - code 6

Opens a file for reading or updating, returning a handle which can be used for getting and putting byte(s).

1/ Client -> File Server (Command port)

```
Bytes 1-5  = Standard TX header
  "   6    = 0 - file need not already exist, and will be created
             1 - file must already exist
  "   7    = 0 - opening for update,
             1 - opening for reading only
  "   8+   = File/dir. title
```

2/ File Server -> Client (reply port)

```
Bytes 1-2  = Standard RX header
Byte  3    = Handle
```

The file will be opened only if the user has sufficient access. A file may be opened several times for reading, but only once for updating (see section 2.10.5). The file will be created with a default size of 1K if:

a) it doesn't already exist
b) it is opened for update
c) the user specified that it need not exist

13.9.7 Shut (CLOSE) -code 7

Closes an open file.

1/ Client -> File Server (command port)

```
Bytes 1-5  = Standard TX header
Byte  6    = Handle
```

2/ File Server -> Client (reply port)

```
Bytes 1-2  = Standard RX header
```

All updated blocks will be written to the file before it is closed.

13.9.8 Get byte - code 8

Reads a single byte from file.

1/ Client -> File Server (command port)

```
Byte  1    = Reply port
Byte  2    = function code (8)
Byte  3    = File handle
```

2/ File Server -> Client (reply port)

```
Byte  1   = command code = 0
Byte  2   = Return code
```

```
     Byte  3    = Byte read or &FF if first byte after file
     Byte  4    = Flag

The flag will be as follows:
     &00 - normal read byte
     &80 - if this is the last byte
     &C0 - if this is the first byte after the end of the file


The sequence number is explained in the next section.
```

13.9.9 Put byte - code 9

Write a single byte to a file

1/ Client -> File server (command port)

```
     Byte  1    = Reply port
     Byte  2    = Function code (9)
     Byte  3    = File handle
     Byte  4    = Byte to be written
```

In addition, the least-significant bit of the 7-bit control code in the Econet control block is a sequence number for the operation (0 or 1).

2/ File server -> Client (reply port)

```
     Byte  1    = command code = 0
     Byte  2    = Return code
```

The sequence number is necessary because it is the file server which remembers the current file position. If the file server gets the sequence number it is expecting, it increments the file pointer by one and increments (i.e. inverts) its copy of the sequence number. If the file server receives the wrong sequence number, it assumes that this is a repeat of the last operation and so does not change either the file pointer or the sequence number. The file server returns the sequence number in the control byte of the control block.

After opening a file the first sequence number is expected to be zero.

The client should be prepared to repeat a call if it gets no reply, or one with the wrong sequence number. The client must increment its sequence number when it has successfully completed an operation.

13.9.10 Get bytes and Put bytes - codes 10 and 11

Read/write a group of bytes

1/ Client -> file server (command port)

```
     Byte  1    = Reply port
     Byte  2    = Function code (10 or 11)
     Byte  3    = Data/acknowledge port
     Byte  4    = File handle
     Byte  5    = Flag - 0 to use given offset,
                         non-zero to use sequential pointer
```

```
Bytes 6-8  = Number of bytes to transfer
Bytes 9-11 = Offset in file if required
```

There is also a sequence number held in the least-significant bit of the Econet control block's control code.

2/ File server -> Client first reply

```
Byte  1    = Command code = 0
Byte  2    = Return code
Byte  3    = Data port (put byte only)
Bytes 4-5  = Maximum block size written (put byte only)
```

There then follows a data transfer as for LOAD and SAVE. If the file size is zero, the data transfer steps are not executed.

3/ File server -> Client second reply

```
Byte  1    = Command code = 0
Byte  2    = Return code
Byte  3    = Flag - 0 if all O.K., &80 if this read includes last
             byte (always zero for putbytes).
Bytes 4-6  = number of bytes actually transferred
```

The client must maintain a sequence number as for Getbyte and Putbyte.

13.9.11 Read random access info - code 12

1/ Client -> File server (command port)

```
Bytes 1-5  = Standard TX header
Byte  6    = file handle
Byte  7    = ARG
             ARG=0 - read file pointer
             ARG=1 - read file extent
             ARG=2 - read file size
```

2/ File Server -> Client (reply port)

```
Bytes 1-2  = Standard RX header
bytes 3-5  = Information reutned (lo byte first)
```

13.9.12 Set random access info - code 13

1/ Client -> File server (command port)

```
Bytes 1-5  = Standard TX header
Byte   6   = File handle
Byte   7   = ARG
             ARG=0 - set sequential pointer
Byte 8-10  = Value to be set (lo first)
```

2/ File server -> Client (reply port)

```
Bytes 1-2 = Standard RX header
```

13.9.13 Read disc info - code 14

1/ Client -> file server (command port)

```
Bytes  1-5 = Standard TX header
Byte   6   = first drive number to get info on
Byte   7   = number of drives to get info on
```

2/ File server -> Client (reply port)

```
Bytes  1-2 = Standard RX header
Byte   3   = number of drives found
Byte   4   = Drive number of first drive requested
Bytes  5-20= Disc name on drive padded with spaces
Byte  21   = Drive number of second drive requested
Bytes 22-37= Disc name on second drive
Bytes 38+  = Info on remaining drives in same format
```

Generally only two drives are present, but the interface will cope with any number.

13.9.14 Read logged on users - code 15

1/ Client -> File server (command port)

```
Bytes 1-5  = Standard TX header
Byte  6    = First user entry to get info on
Byte  7    = number of user entries to examine
```

2/ File server -> Client (reply port)

```
Bytes 1-2  = Standard RX header
Byte  3    = number of users found
Bytes 4-5  = machine number where user logged on (lo then hi)
Bytes 6-15 = User id. padded with spaces
Byte  16   = privilege of user - 0 unprivileged,
             non-0 if privileged
Bytes 17+  = Further entries in same format
```

Note that the number of users found will be zero if the first user entry to be examined is greater than the number of users logged on.

13.9.15 Read data and time - code 16

1/ Client -> File server (command port)

```
Bytes 1-5  = Standard TX header
```

2/ File server -> Client (reply port)

```
Bytes 1-2  = Standard RX header
Bytes 3-4  = Date, standard format:
             byte 3 - days
             byte 4 (lo bits) = month
             byte 4 (hi bits) = years since 1981
Bytes 5-7  = 0 if no time-board attached to file server
             byte 5 - hours
             byte 6 - minutes
```

```
                byte 7 - seconds
                if time board attached
```

13.9.16 Read "end of file" status - code 17

```
1/ Client -> file server (command port)

     Bytes 1-5  = Standard TX header
     Byte  6    = Handle of file

2/ File server -> Client (reply port)

     Bytes 1-2  = Standard RX header
     Byte   3   = &FF if file pointer is outside file
                  &00 otherwise
```

13.9.17 Read object info - code 18

```
1/ Client -> file server (command port)

     Bytes 1-5 = Standard TX header
     Byte  6   = ARG
                 ARG = 1 - Read creation date
                 ARG = 2 - read load and execute address (8 bytes)
                 ARG = 3 - read size (3 bytes)
                 ARG = 4 - read type/access byte
                 ARG = 5 - read all file attributes
                 ARG = 6 - read access/cycle/dir. name of given dir.
     Bytes 7+  = Object name (terminated by CR)

Note - format of access/type byte:
bits 0-1 = public RW
bits 2-3 = owner RW
bit 4    = locked
bit 5    = type (1 = dir., 0 = file)

Result from file server depends upon ARG:

ARG = 1-5

File server -> Client (reply port)

     Bytes 1-2  = Standard RX header
     Byte   3   = 0 if object doesn't exist
                  &FF if object does exist
     Bytes 4+   = results (in order load/execute/size/access/date)


ARG = 6

File server -> Client (reply port)

     Bytes 1-2  = Standard RX header
     Byte  3    = Unused
     Bytes 4-5  = 0 and 10 (used by BBC Micro NFS only)
     Bytes 6-15 = Dir. name padded with spaces
     Byte  16   = Access to dir. (0 - owner, &FF - public)
```

```
     Byte  17   = Cycle number of dir.
```

13.9.18 Set file info - code 19

```
1/ Client -> file server (command port)

     Bytes 1-5  = Standard TX header
     Byte   6   = ARG
                  ARG = 1 - set load/execute/access
                  ARG = 2 - set load address
                  ARG = 3 - set exec. address
                  ARG = 4 - set access byte
     Bytes 7+ = details to set (size depends on ARG)
     Bytes n+ = file name (terminated by CR)

2/ File server -> Client (reply port)

     Bytes 1-2  = Standard RX header
```

13.9.19 Delete object - code 20

```
1/ Client -> file server (command port)

     Bytes 1-5  = Standard TX header
     Bytes 6+   = Object name (terminated by CR)

2/ File server -> Client (reply port)

     Bytes 1-2  = Standard RX header
     Bytes 3+   = load/exec/size
```

13.9.20 Read user environment - code 21

```
1/ Client -> file server (command port)

     Bytes 1-5  = Standard TX header

2/ File server -> Client (reply port)

     Bytes  1-2 = Standard RX header
     Byte     3  = length of disc name
     Bytes  4-19= Currently selected disc name, padded with spaces
     Bytes 20-29= CSD name, padded with spaces
     Bytes 30-39= Library name, padded with spaces
```

13.9.21 Set user option - code 22

```
1/ Client -> file server (command port)

     Bytes  1-5 = Standard TX header
     Byte   6   = New boot option (lower 4 bits)

2/ File server -> Client (reply port)

     Bytes  1-2 = Standard RX header
```

13.9.22 Log-off - code 23

1/ Client -> File server (command port)

Bytes 1-5 = Standard TX header

2/ File server -> Client

Bytes 1-2 = Standard RX header

13.9.23 Read user info - code 24

1/ Client -> file server (command port)

Bytes 1-5 = Standard TX header
Bytes 6+ = User name, terminated by CR

2/ File server -> client (reply port)

Bytes 1-2 = standard RX header
Byte 3 = User privilege (0 - unprivileged, non-0 - privileged)
Bytes 4-5 = Machine number logged on (lo byte first)

An error will be returned if the user is not logged on.

13.9.24 Read File Server version number - code 25

1/ Client -> File server (command port)

Bytes 1-5 = Standard TX header

2/ File server -> client (reply port)

Bytes 1-2 = Standard RX header
Bytes 3+ = Character string describing version, terminated by CR.

Appendix A - Summary of the Filing System

This appendix is intended to be a quick overview of the file server for people who have some experience of filing systems, and who wish to get a general idea of the capabilities and style of the ACORN file server.

The ACORN File Server provides a hierarchical directory structure, each directory containing entries which may be files, or further directories.

A FILE is simply a copy of a block of memory of variable size, with a load address specifying where it should be loaded, and an execute address, specifying where execution should start if the file is loaded as a program.

In addition to load and execution addresses, the size of the file and the date it was created (last SAVED) are stored in the directory entry.

The file server can support multiple drives, each physical disc on the system being viewed as a complete logical filing system. Each user can select which filing system he wishes to use by specifying the name of the disc he requires (see *SDISC command).

The base of the hierarchy on each disc is the ROOT directory, which is created on the disc when it is initialised. In general each user of a disc will have a directory in the ROOT (the Users Root Directory - URD) which is the base of his own hierachy of files and directories.

Files are referred to by a file title made up of sections separated by ".". Each section corresponds to a level of the hierarchy, the root being referred to by the "$" character. Thus $.JOEY.MYFILE refers to a file (or directory) MYFILE in directory JOEY, which is itself an entry in the root directory ($).

It is also possible to "select" a directory and refer to files from that directory, rather than through the root. For example MYFILE would refer to the same object as mentioned above, if $.JOEY was the selected directory.

When a user logs on or selects a disc, his URD becomes the Currently Selected Directory (CSD) if it exists, otherwise, the ROOT itself becomes the CSD.

Each entry in a directory has an ACCESS field which allows protection from deletion (the LOCK facility), and reading and writing by both other users of the file system, and the directory owner. System privileged users can be created who have owner access to the whole filing system.

Appendix B - List of Commands

Command	Abbrev.	Syntax
*ACCESS	*A.	*A. <file title> <access string>
*BYE	*BY.	*BY.
*CAT	*.	*. <opt. file title>
*CDIR	*CD.	*CD. <file title>
*DELETE	*D.	*D. <file title>
*DIR	*DIR	*DIR <opt. file title>
*DISCS	*DISCS	*DISCS
*EXEC	*E.	*E. <file name>
*I AM	*I A.	*I AM <user id.>
*INFO	*I.	*I. <file name>
*LIB	*LIB	*LIB <dir>
*LOAD	*L.	*L. <file title> <opt. address>
*NEWUSER	*N.	*N. <user id>
*OPT	*O.	*O. m,n
*PASS	*P.	*P. <old pw> <new pw>
*PRIV	*PR.	*PR. <user id> <priv. char>
*REMUSER	*REM.	*REM. <user id>
*RENAME	*R.	*R. <old file title> <new file title>
*SAVE	*S.	*S. <file title> <addr.> <addr.> <opt. addr.>
*SDISC	*SDI.	*SDI. <disc name>
*SPOOL	*SP.	*SP. <file name>
*USERS	*USERS	*USERS

Appendix C - Error numbers

The word 'object' in the following descriptions refers to a file or a directory.

ERR (hex)	Description
13	USRMAN.RESTART called twice
14	Object not a directory
15	User not logged on
16	Machine zero is invalid
21	Cannot find password file
27	Password file syntax error
29	Object "$.PASSWORDS" has wrong type
31	STRMAN.RESTART called twice
32	SIN = 0
33	REF COUNT = &FF
34	REF COUNT = zero
35	File too big or zero length
36	Invalid window address
37	No free cache descriptors
38	Window ref count > 0
39	Big buffer already in use
3A	Invalid buffer address
3B	Ref. count = &FF
3C	Store deadlock
3D	Arith. overflow
42	Broken directory
46	Attempting to set WR attributes to a directory
48	Not defined
4C	No write access
4E	Too many entries asked for in EXAMINE
4F	Bad ARG to EXAMINE.
53	SIN not for start of chain
54	Disc not a file server disc
55	Both sector maps corrupt
56	Illegal drive number
57	Map seq. nos. differ by >1
58	Object size zero
59	New map doesn't fit in old space
5A	Disc of same name already in use
61	RNDMAN.RESTART called twice
64	All handles open - handle table full
65	Object not open
66	Cannot copy file handles
67	RANDTB full
69	Object not a file
6D	Invalid arg to RDSTAR
6F	GETBYTE; byte after last in file
71	Invalid number of sectors
72	Store address overflow
73	Accessing beyond file end

74	Invalid SIN
83	Too much data from client (SAVE)
84	Wait bombs out
85	Invalid function code
86	Undefined
8A	File too big
8C	Bad privilege letter
8D	Excess data in PUTBYTES
8E	Bad INFO argument
8F	Bad arg to RDAR
A0	Line jammed
A1	Net error
A2	Not listening
A3	No clock
A4	Bad TXCB
A5	No reply
AD	Mode x
AE	User not logged on
AF	Types don't match
B0	Renaming across two discs
B1	User id. already exists
B2	Password file full
B3	Maximum directory size reached
B4	Directory not empty
B5	Trying to load a directory
B6	Disc error on map read/write
B7	Attempt to point outside a file
B8	Too many users
B9	Bad password
BA	Insufficient privelege
BB	Incorrect password
BC	User not known
BD	Insufficient access
BE	Object not a directory
BF	Who are you?
C0	Too many open files
C1	File not open for update
C2	Already open
C3	Entry locked
C6	Disc full
C7	Unrecoverable disc error
C8	Disc number not found
C9	Disc protected
CC	Bad file name
CF	Invalid access string
D6	Not found
DE	Channel
DF	End of file
FD	Bad string (filename etc.)
FE	Bad command

Appendix D - BBC Microcomputer Machine Operating System calls

This appendix lists the machine operating system calls used specifically with the Econet filing system.

OSBYTE calls

Call Hex	Decimal	Function
32	50	Poll transmit block
33	51	Poll receive block
34	52	Delete receive block
35	53	Sever REMOTE connection

OSWORD calls

Call Hex	Decimal	Function
10	16	Call TRANSMIT
11	17	Open/read receive block
12	18	Read remote routine arguments
13	19	Read/set miscellaneous net information
14	20	Call file server/remote machine

EVENTS

Number	Event
8	Remote procedure call

ACORN COMPUTERS LIMITED
Fulbourn Road, Cherry Hinton
Cambridge. CB1 4JN
Telephone (0223) 245200