Longman Micro Info

THE ECONET MICRO GUIDE

Christopher Dawkins

7·95

Longman Micro Guides

THE ECONET MICRO GUIDE

Christopher Dawkins

Series editor: Bryan Spielman

Longman

London and New York

British Library Cataloguing in Publication Data

Dawkins, Christopher
The Econet micro guide.—(Longman micro guides)
1. BBC Microcomputer 2. Computer networks
I. Title
001.64'24 QA76.8.B35

ISBN 0-582-36215-6

Library of Congress Cataloging in Publication Data

Dawkins, Christopher.
The Econet micro guide.

(Longman micro guides)
Includes index.
1. Econet (Local area network system) I. Title.
II. Series.
TK5105.8.E25D39 1985 001.64'404 84-25075
ISBN 0-582-36215-6

LONGMAN GROUP LIMITED
Longman House, Burnt Mill, Harlow, Essex CM20 2JE, England
and Associated Companies throughout the World
Distributed in the United States by Longman Inc, 1560 Broadway, New York, NY 10036

First published 1985
ISBN 0 582 36215 6

Printed in Great Britain by
Butler & Tanner Ltd,
Frome and London

Longman Micro Guides

The microcomputer is becoming as much a part of life as the motor car. Its use in schools, in business, in industry and in the home is increasing fast. Its power and the extraordinary variety of the faces it can assume make it unlike any commonplace apparatus we have had before. Thus, coming to terms with it - as everyone must surely do sooner or later - can be more than usually demanding. But it can be more than usually enriching too. Longman Micro Guides are intended to give a helping hand over some of the hurdles. They are handbooks, essentially practical.

The aim has been to keep them direct, clear and friendly. Jargon is either avoided or, where it is necessary, explained. Although they are brief they include enough detail to enable a reader to put the advice they contain into immediate practice.

The authors are all experts in their field and write from experience.

Other Longman Micro Guides

The BBC Micro Guide
Connecting Computers to Peripherals
The Electron Micro Guide
Micros in Control

Microcomputers in Early Education
Microcomputers and Teaching Biology
Microcomputers and Teaching Economics
Microcomputers and Teaching History

Contents

Appendices

Preface

In the 1950s, those people without their own computer had to use the postal system to send work to machines they needed to use. In the 1960s telephone access was developed, universities started sharing their computers with each other and companies started selling computer time to remote users. In each case there was only one computer to consider: all access was from 'dumb' terminals. In the 1970s institutions started acquiring a second or third machine, and in the 1980s they order computers by the dozen. Once one has more than three or four computers there arises a compelling need to connect them together, largely for the economies of resource sharing. Thirty years ago the computer was expensive and peripherals were cheap, so network systems concentrated on sharing one computer between many peripherals. Now the computers are cheap and peripherals are expensive, so networking concentrates on sharing the peripherals between many computers. Universities started developing multiple-computer networks in the 1970s, and when microcomputers started being sold at more than one or two per institution it became imperative to interconnect them. Every serious microcomputer on the market now has its own network system: some have several alternative networks and some networks can accommodate a variety of computers. This book describes the Econet system for computer networking as implemented on the BBC microcomputer, and mentions alternative networks and alternative machines.

What are the advantages of putting machines on a network – compared with, for example, providing each machine with its own disk drive? First, the disadvantages: (a) machines with disk drives are much more portable and (b) if a network goes wrong all the machines are affected. Portability can however be a disadvantage, and there is not a lot to go wrong with a properly installed network – there are no moving parts. The advantages? First, the ability to share a common filestore: all stations share the same files which leads both to economies on disk space and ease of updating. Second, communications: for example, electronic mail between stations. Third, the multiplication effect: printserving, for example, multiplies the use of a printer by the number of stations on the network. An institution need buy only one teletext adaptor: every station on the net can then use it. The first advantage is the main justification for a network and some networks are limited to it, but there are many ways the other two can benefit an institution.

There is a lot of new jargon in this subject. In most cases, this is explained where first encountered – in others, it is explained in the glossary at the end. Some alternative terms are used interchangeably throughout the book (for example, a 'buffer' is a 'memory area' associated with an input/output operation). Decimal and hexadecimal numbers are freely mixed, but those in hexadecimal are preceded by an ampersand (&).

Acknowledgements

This book is really a collation of things I have learnt from those pupils who have learnt about networks and installed and operated them for me at Felsted: at my age it is just not possible to keep up with things. My technique is to provide the most expert pupils with manuals, equipment, tea and creme eggs and await results. Among many miracles that have been wrought have been the processing of a megabyte data file on a computer of 1958 vintage in 1980, interfacing of a Winchester and CP/M 2.2 to a Nascom in 1981, networking BBC machines to a Level 2 fileserver in 1982 and fileserving from the Winchester in 1983. In 1980 I could not have written a word of this book: little did he know how little I knew when Bryan Spielman asked me in 1981 to write it. However, three years of incredible patience from Bryan has given me time to learn enough from Andrew Gordon and David Bisset to assemble the pages that follow. My debt to Andrew and David is incalculable, but I have also learnt from others who have worked on the Econet at Felsted. Stephen Lawrence did much work on the Apple, Nascom-1 and CP/M Econet, Stuart Page processed the megabyte file that is now on the network, James Conder wrote much Admin software, David Salter produced the security system, and Colin Stark, Anthony Engeham and David Deacon wrote the utilities that make our Econet so viable. Outside Felsted I have learnt from Joe Dunn and Jes Wills at Acorn, Robin Newman at Oundle and Bob Napier of Barson Computers.

While I have taken the advice of many people in writing this I have forgotten much of it (old age) and ignored a little (Andrew, for example, strongly and correctly disapproves of poking and peeking in the BBC computer, but I find it makes life easier). The useful information in this book comes entirely from the gentlemen above; the faults come entirely from myself.

Christopher Dawkins

1 Networks

Computers process information. For most applications, the value of the computer system depends on the quantity, the relevance and the accessibility of the information in the system. There must be enough information, it must be the right information and it must be available. Unfortunately information storage is more expensive than computer power, especially if a separate copy of the information has to be stored for each machine that might need it. Both the sharing and the timely acquisition of information require a computer to be in communication with an information source.

1.1 Communications systems

These require

a a transmitting station or stations;
b a receiving station or stations;
c a communications medium between the two;
d an agreed code; and
e a system for making sure that correct information is received by the correct station.

The problem of addressing the information to the right place is solved in various ways by various systems. The Econet system is Ethernet type, which means it is an isochronous packet-switched SDLC STDM baseband LAN using bus topology with CSMA/CD, a democratic protocol and a bidirectional twisted-pair balanced-line medium with independent clock and data, although anyone describing it thus ought to be shot - and not only because some of the descriptions above imply each other.

The two broad categories of communications system are

a broadcast, where the receiver must distinguish between messages addressed to it and messages addressed to others; and
b directed, where the medium (e.g. the ordinary telephone system) directs the message to the correct recipient.

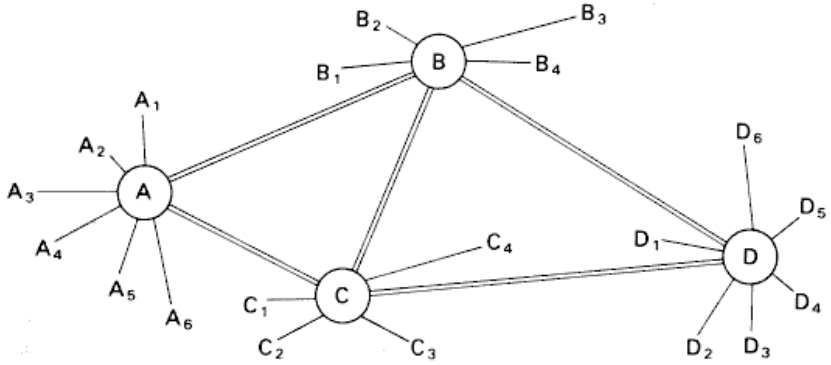

Figure 1.1 Star network

The three main ways of arranging the cables for a communications system are star, ring and bus.

1.2 Star networks

The telephone network consists of a number of interconnected stars, sometimes called a *mesh*. Although modern trunk lines multiplex many conversations onto single physical cables, the system still works on the principle of one cable from the 'master' to each terminal. Such networks may be circuit-switched or packet-switched. Traditional telephone networks are circuit-switched, i.e. a special circuit is set up between each pair of communicating terminals on the network for the whole duration of each conversation. In the early 1970's packet-switching was developed for data transfer on some telephone systems: the conversation is broken up into packets which are independently routed through the network between the stations. Packet-switching allows much more efficient use of the cables – it is the job of the exchanges to route each packet the most economical way to its destination. Packet-switching became economic only as electronic intelligence became cheaper than copper cables. It uses the intelligence of the exchanges to compensate for shortcomings in the medium. In ring systems the repeaters on the ring provide the necessary 'intelligence': in bus systems the cable is entirely passive and all the intelligence has to be in every station. Star networks can be designed with or without packet-switching – rings and buses must communicate in packets. Consider the simple star network in Figure 1.1.

There are four possible routes between A and D and three possible routes between any other pair of stations in Figure 1.1. With circuit-switching, between three and five conversations could take place – e.g. A_1/D_1, A_2/D_2 and B_1/C_1. These three would monopolise the trunk routes (assuming each can only take one conversation). If conversations are not continuous, packet-switching could allow A_3/D_3 and A_4/D_4 to converse as well, their packets using the cables during the lulls (which may be very short) in A_1/D_1 and A_2/D_2. With packet-switching, each conversation is broken up into many small packets which are numbered and sent sequentially. The route is chosen independently for each packet. Consider a message being sent from A to D. Packets leaving A will choose the least congested route at the moment of departure – which could be via B or via C. On arrival at B or C they may decide their onward route anew. Clearly if BC is unoccupied a packet may spend some time shuttling to and fro waiting for an opening BD or CD; the logic may forbid or discourage this – or keep copies at both B and C and leave D to eliminate any duplicates that may arrive – or even in some cases throw away packets that cause congestion and rely on the recipient to request a retransmission from the sender.

1.3 Ring networks

The stations are arranged on a ring of cable. Signals have to pass through all stations in turn – though in most systems they pass through *repeaters* which are connected to the computers through special *access boxes*. Rings have some electrical advantages because the signal in any wire travels in one direction only. This helps in

a fault-finding, because each particular piece of wire carries a signal from one particular station to the next, so a fault can be immediately localised;

b machine protection, because it is easy to put in optical isolation to protect machines from (for example) lightning strikes; and

c amplification, because amplifiers cannot easily amplify in both directions.

A ring need be neither circular nor regular so long as it remains topologically equivalent to a circle. The traffic on the ring must be in packets (or *frames*); these each have a header which gives the addresses of the sender and the recipient. The monitor station initiates and maintains checks on the stream of packets. Some ring systems manage without a monitor station.

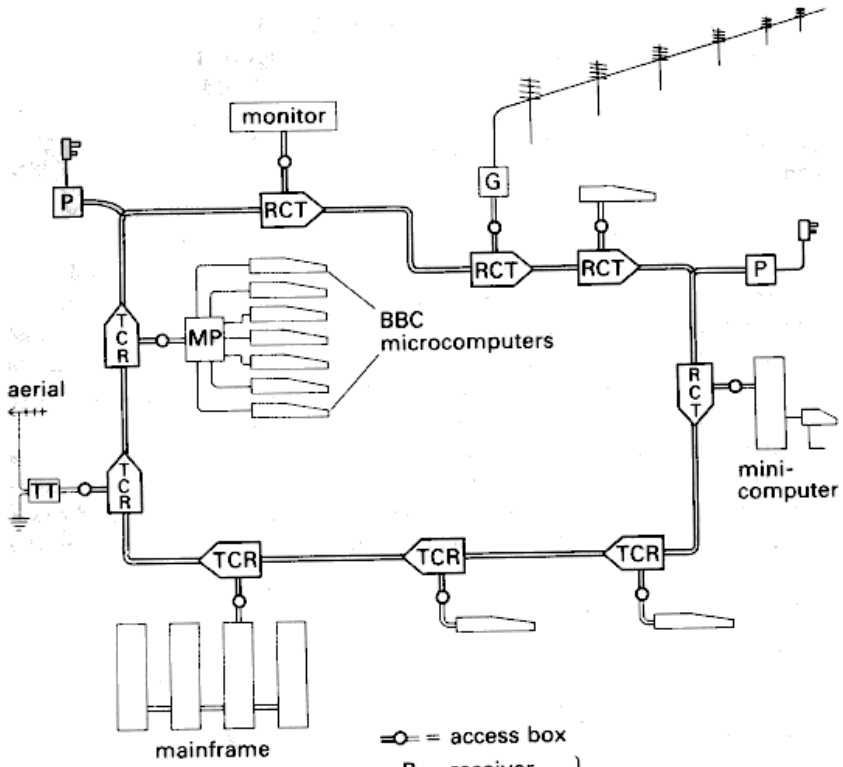

Figure 1.2 Ring network

1.4 Bus networks

A bus system is inevitably a broadcast one: the Econet is a bus system. One cable connects all machines in parallel. Figure 1.3 shows a bus network. It is the simplest to draw and the simplest to operate in that stations may be plugged in and unplugged anywhere at any time without affecting the network, although it is more difficult to locate a fault in the cable. Other problems are that T-junctions are difficult, length is more limited than a ring and proper termination is critical.

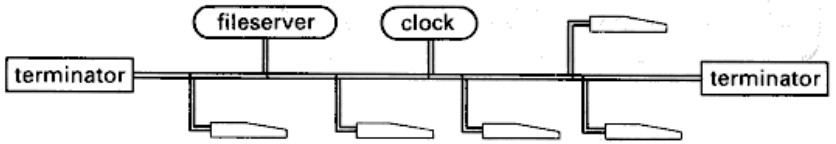

Figure 1.3 Bus network

Since only one message can be on the bus at any one time, the machines sharing it need to cooperate. To do so, each station needs

- **a** an address – in the Econet this is an 8-bit number hard-wired into each station with 8 small connectors, each of which may be present or absent. Up to 256 values can be specified in 8 bits: zero and 255 are reserved for broadcasts, so stations can be numbered between 1 and 254 inclusive;
- **b** an ability to read information from the bus;
- **c** an ability to put information onto the bus;
- **d** an ability to detect when the bus is free (since only then may it transmit);
- **e** an ability to detect a *collision* between two or more stations;
- **f** a system (*collision arbitration algorithm*) to ensure that two stations that have collided back off for different lengths of time before retransmitting;
- **g** a method for checking correct receipt; and
- **h** a system (*protocol*) for arranging retransmission of packets corrupted in transit. Note: this is not the same thing as retransmission of a packet that could not be transmitted owing to a collision.

All complicated, but complex logic is now a cheap commodity, especially when reproduced in large quantities. In practice it is sometimes possible to dispense with some of the features above because, if every feature is present, we are taking great pains to prevent data corruption and then taking steps to deal with it if it occurs. The first Econet had no collision arbitration and worked well. Later versions may dispense with the collision detection system (to save components on the interface) and rely on data checking and retransmission to deal with any problems that may arise. They will work, but perhaps not so well on heavily loaded networks.

2 Development of the Econet

2.1 Issue 1

Issue 1 of the Econet system was devised by Acorn in 1980 for their range of *Atom* and *System* microcomputers. It worked remarkably well despite having no collision arbitration algorithm. Acorn had originally planned to implement a ring - a micro version of the Cambridge University 'Ring' which was then making computer headlines. However, the 'Ethernet' system (developed originally in Hawaii to allow island-based computers to communicate with each other on one radio channel) was earning its share of the headlines, and Acorn decided a cheaper network could be developed using the bus approach of the Ethernet but with some trimming of its specifications. The prefix *Eco* was chosen to mean 'local' or 'economical': the latter is nearer the Greek 'oikos' which means 'home' - so 'Eco' actually means 'homely'.

2.2 Issue 2

Issue 2 occupied much of Acorn's effort during 1981. Its main feature was abolition of the acknowledge packet: instead, the receiving machine simply held the line (transmitted idles) for a certain number of clock cycles if reception was satisfactory. The transmitting machine listened to the line immediately after transmission, and retransmitted the packet if it did not detect these idles. This protocol halved the number of packets on the net and thus the number of interrupts to every machine. It made everything faster but was just not reliable enough especially over long or heavily loaded nets. It was one of those bright ideas which work well in the lab but not in the field.

2.3 Issue 3

Issue 3 (the BBC system and that for Atoms, Systems, and all other Econet implementations since 1982) has the full packet acknowledge of version 1, with 16 bits for the station ID (the higher 8 select other networks connected via gateways) and 32 bits on all memory addresses (to

allow for the 4.3 gigabyte address range of the second processor). There was a 'privileged machine' concept until 1984, but this was eliminated because pupils capable of altering a machine's number were using it to make their machine privileged and thus able to PEEK passwords from protected machines (such as Fileservers). The network is now as secure as a broadcast system can be: since all messages are necessarily received by all stations, it is possible for anyone capable of writing low-level network software to read any message and, in theory, to decode it.

2.4 The next issue

The major improvement in the next issue of the network is buffering of file I/O. The slowness of SPOOLing and EXECing on issue 3 is its most serious fault: all file I/O using BPUT#, BGET#, PRINT# and INPUT# is equally slow. This is because all these operations are sent to the operating system byte by byte, the OS sends them to the NFS byte by byte and the NFS puts each byte on the network separately. Each byte requires 2 packets (8 'frames') to be exchanged with the fileserver, involving a total of 42 bytes. *SPOOLing 1K of data requires 10240 interrupts to every machine on the network and 42K bytes to be transmitted, while *SAVEing the same data requires only 10 interrupts and 1066 bytes. Some buffering in the client is clearly needed, and is being put into the next issue. There is a snag of course - more RAM is needed so PAGE goes up.

2.5 Further development

Further development will of course take place, as bright ideas on improvements are occurring all the time. The only problem for future developments is the need (so far as it exists) to remain compatible with all the systems already operating. There has already been some publicity for the BBC Ring interface, which being of a different nature will not be directly compatible with the Econet. A ring makes much more sense than a bus for long distance work (around a site), partly because it can run faster but largely because faults can be instantly traced. When all the machines are in one room it is easy to locate a fault by unplugging them all and plugging them in again one by one. When they are spread over half a mile of corridors fault-tracing involves a lot of leg-work. Most current rings use wire of a substantially lower specification than the Econet cable: cable laid

for a bus can later be used for a ring though (depending on whether a two- or a four-wire system is chosen for sending the signals) it will probably be necessary to add a second cable for the return signal. Most ring systems do not put the client computers actually on the ring. Instead there is a *repeater* on the ring and an *access box* connected to the repeater (these two can be combined on one circuit board). There may be a dozen computers sharing the same access box, and in the case of BBC machines these would use Econet for local communications with it. Conversion from Econet to ring is thus unlikely to affect the BBC machine or its immediate net at all. It will merely improve the longer-distance communications and make the system manager's job much easier.

3 The Econet system

3.1 The functions of the network

These are

a to provide shared mass-storage facilities on a 'fileserver' or two;
b to provide communications: 'electronic mail' and similar applications; and
c to 'multiply' access to other facilities that may already exist for a single machine, such as printers, plotters, teletext adapters, meteorological stations, telephone lines and burglar detection devices.

3.2 The components of an Econet system

a The network cable, with sockets. The cable must be topologically a line: sockets may be in any number and any place, though connections must be good. The cable should have five wires in it, being one twisted pair for *clock*, one twisted pair for *data* and one single wire for earth. All sockets on it are simply wired in parallel, so all signals go equally and nearly simultaneously to all machines. A building equipped with Econet will normally have rather more sockets than computers, for when the cable is laid it is easy to put sockets in at intervals wherever a computer may later be required. These are 5-pin 180° DIN sockets such as those on most cassette recorders. Econet-equipped computers have a similar socket and are plugged into the nearest wall socket with a standard lead. You can unplug computers anywhere at any time, carry them somewhere else and plug them in again – the network will not be affected.

b Two terminators, one at each end of the cable. These need power, but if the clock is designed to put power onto the cable then the terminators can draw their power from it. Terminators need be no larger than a thimble.

c A 'clock', which puts a 5-V square wave of frequency between 100 and 300 kHz onto the clock lines of the cable. All signals on the cable are synchronised by the clock, which is why this sort of network system is generally described as SDLC (Synchronous Data Link Control).

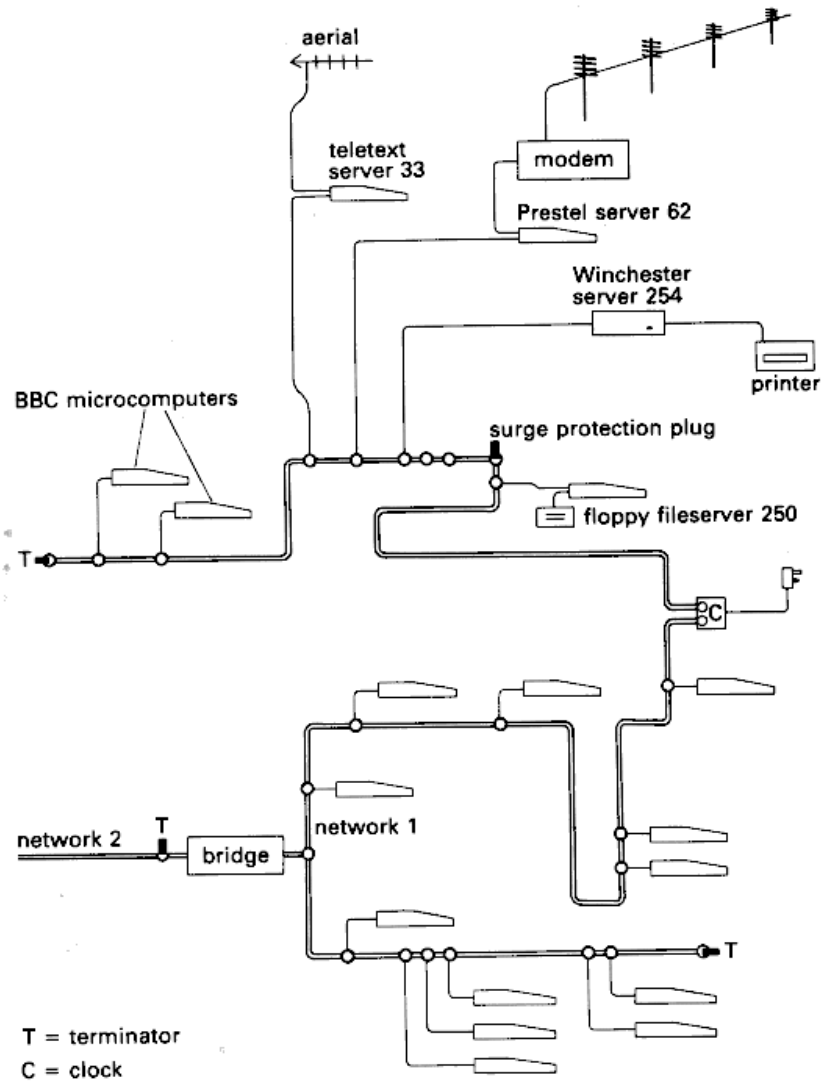

Figure 3.1 The complete network

Because data is only recognised by stations if it occurs when the clock is up (called *mark*), it can help on long cables to give the clock a longer up time than down (called *space*), and better clocks provide the facility

to alter the mark/space ratio. The words mark and space derive from telegraphy a century ago, when the high voltage caused a mark to appear on a paper roll.

d The computers using the network, generally described as *clients*. These, currently, may be Acorn machines (BBCs, Atoms, Systems 3, 4 or 5) or CP/M systems (Torches, 380Zs, 480Zs or S-100 machines using commercially available interfaces, and Nascoms or Apples using home-made lash-ups). Econet interfaces may be developed for other common microcomputers.

e The Fileserver. Access to a central program/data library on a fileserver is one of the main reasons for having a network in the first place.

f A Printserver, which in most cases can be the same machine as the Fileserver – and in others can be a machine that also has another function. An advanced printserver will have spooling facilities, so that it can handle more than one station at once.

g Other fileservers. There are fileserver programs that can run on any disk machine to convert it into a network server, so its disks become available to any other machine on the network. In some cases the machine is then dedicated to serving, and cannot be used for anything else: in others the serving continues in background mode so the machine can be used for other purposes while being a fileserver.

h Other printservers, so as to provide network access to a variety of printers or plotters.

i Other servers, e.g.
- (i) a Teletext server, to allow all machines access to teletext pages,
- (ii) a Prestel server, or
- (iii) a meteorological station.

j A store-and-forward message system (a *mailserver*), so that anyone from any station can send message to anyone else even if they are not currently logged in. The autoboot option on the fileserver can be organised to provide mailserving facilities.

k A network monitor, or Big Brother, to keep an eye on traffic.

l A 'noticeboard' system, to function rather as airport displays but with notices that can be supplied from and read at any station.

m Any other facility that all stations might wish to share.

3.3 Logging in

A BBC computer will come on already in the NFS (Network Filing System) if the internal arrangements are set appropriately, but if it does

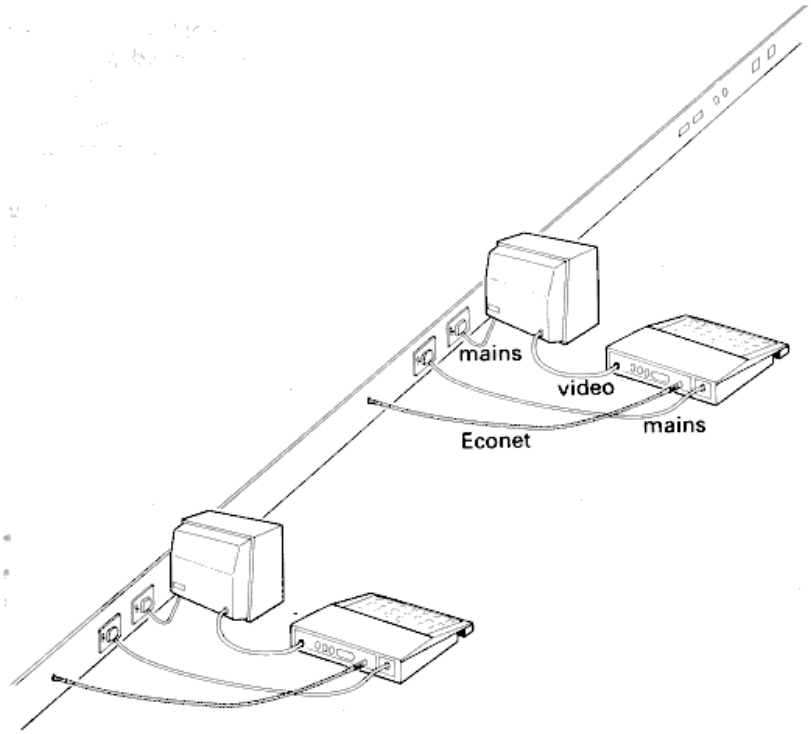

Figure 3.2 On the Econet

not you can use the command *NET. You can use this command even if a Basic or other program has already been loaded, unless it has relocated itself into the Econet workspace. In general, the first thing to do when you approach the network is to let the fileserver know who you are with the

*I AM FRED command

if 'FRED' is the name by which you are normally known to the system. Most systems would allow public users to log in as 'USER' or something similar.

The power of the Econet system depends now largely on what the fileserver can provide for your machine. If you type *ABCD the text 'ABCD' will first be passed around the ROMs in your machine. If

none recognise it (you can bypass the ROMs by typing *RUN ABCD or */ABCD), it will be passed onto your current fileserver, which is unlikely to recognise it as a command. It will then search your directory for a file 'ABCD', and if it finds it it will load it into your machine and run it. If there is no ABCD in your Currently-Selected-Directory your Currently-Selected-Library will be searched (normally a directory called $.LIBRARY), and only if there is no ABCD there will you get 'Bad Command' back at your station. So extra commands can be provided for every machine on a network by placing a suitable file in $.LIBRARY on the fileserver, which is why the fileserver is so crucial to the power of a network. The Econet system has some extra flexibility in that it allows any machine to access any number of file (and other) servers, as well as to communicate with any other machine or machines on the network without involving a fileserver. Much of the remaining discussion in this chapter presumes you have a Level 2 fileserver with a generous supply of command files, and a network with well developed supporting equipment.

On logging in, you may be informed of any waiting 'mail' if the system has been set up to do this. Once this is over, you can inspect your own directory with the command

CAT (short for 'CATALOGUE'. It can be abbreviated further to a dot: '.' will have the same effect)

or you may consult other directories such as the central library (with the command *CAT $.LIBRARY). Basic programs can then be loaded with

LOAD "Progname"
or saved with
SAVE "Progname"

and you or your programs can access any public data files that may exist.

3.4 Other facilities

If an appropriate command file is available on the fileserver, messages can be sent to other users with

*NOTIFY followed by destination and message

or perhaps

*VIEW

may be used to take a copy of the screen of any other station, or

*REMOTE

may be used, if you have been given the necessary authority, to take over remote control of any other station.

Further facilities depend on what your local system may provide: for example

*LISP

may load a LISP interpreter into your machine from the fileserver thus converting your machine from being a BASIC computer into a LISP one,

*FORTH

may similarly convert your machine to FORTH,

*MAIL (or *MESSAGE)

may allow you to read your mailbox and write to others or

*NB

may bring you the current 'noticeboard' display.

*CONVERSE

may arrange a multi-way conversation between a group of stations,

*TT

may give you access to teletext pages if there is a teletext server on the network,

*PRESTEL

may similarly give you access to Prestel,

*TIME

may give you the current time and date, read either from a real-time clock somewhere or from a teletext page, and

*TEMP

may give you the current outside temperature, if there is a meteorological server.

If there is, as there should be, a printer somewhere on the network you can send output to it either with CTRL/B and CTRL/C (anything output between these goes to the printer) or by using special programs from the library to print (for example) high-resolution-graphics screens or teletext/Prestel pages. Before the CTRL/B is used it is of course necessary to select the network printer (since the default printer for any BBC machine is a local parallel device). There may be several ways of doing this, as there may be several alternative network printers, but the basic command is *FX 5,4 and a common alternative is *PS followed by the number or type of the printer you wish to use. If you are doing graphics dumps with NFS 3.34 in your machine you may have problems because this NFS inserts additional nulls at the end of every buffer. Such nulls rarely affect text printing, but can be disastrous for graphics.

When you have finished, the

*BYE

command will log you out.

4 The Fileserver

The Fileserver is the most valuable part of the network; it is for most purposes its raison d'être and although it is very easy to use it repays close study. Its basic function is to provide a filing system for all stations on the Econet: to allow all stations to save files onto disk and load files from disk. There are at least half a dozen fileservers available for Econet systems: in Acorn terminology they can be classified as 'Level 1' or 'Level 2'. A Level 1 Fileserver is a program (often written in Basic) which allows normal Acorn disks on an ordinary BBC machine to be accessed by anyone on the network. The main limitation of the Acorn version is that it does not allow reading and writing data files. Other problems are that it uses the DFS catalogue structure on the disk (31 files per side), limits file sizes to around 14K and does not allow concurrent print serving. Versions of it – and other Level 1 fileservers – are available that circumvent some or all of these limitations.

4.1 Features of Level 2 fileservers

Level 2 fileservers are very different and are currently offered by Acorn, SJ Research and GSL: most have the following important features.

a Random-access files accessible from all stations.
b A hierarchical directory structure.
c A password system to control access to files and other facilities.
d An unlimited (within reason) number of files per disk.
e A save date (and time, on the SJ fileserver) associated with each file. The GSL fileserver does not have this feature. It allows old files to be purged and incremental backups to be done. An incremental backup means running a program that only transfers to backup media files that have been saved since the last backup.
f A facility to lock files to prevent accidental deletion.
g Independent control for each user of his own and of public access to his files. It may seem odd to provide a facility for a user to deny himself read access to his own files, but there are occasions when it is useful. Much more necessary is the ability to control who else can read or write to them, and all Level 2 systems provide, each in a different way, this facility.

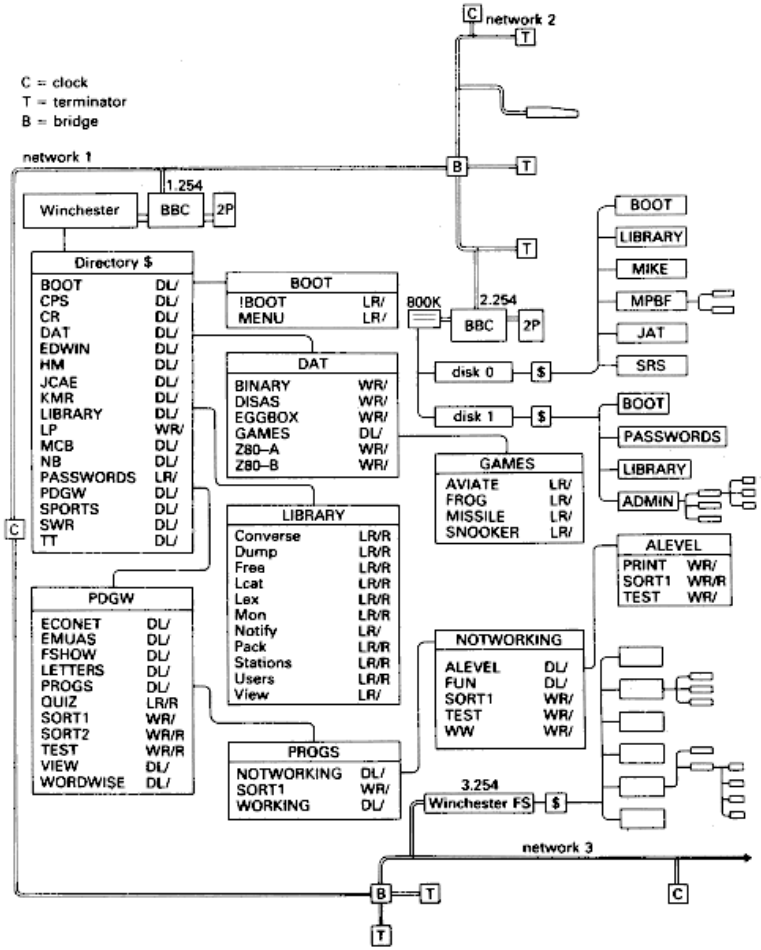

Figure 4.1 The hierarchical network structure

h A library directory, additional to the user's directory, which is automatically open to all users. This is particularly suitable for the implementation of extra commands as, in common with many systems, a command unknown to the operating system is assumed to be the name of an executable file.
i Long file names: nine to fifteen characters depending on the system

- but generally no file 'extensions'. However, CP/M type extensions can be implemented using subdirectories;

j A multiple read facility for any file. Multiple writing is difficult to organise on any system - consider what might happen to a file if two users at different locations are attempting to edit the same part of it at the same time. Most fileservers lock out a complete file to other stations if anyone is writing to any part of it, but some include locking out at the sector level so that different stations can update different sections of a file. Multiple updating of a file can be very useful, but it is only when one considers in a detailed manner how it might operate that one realises how difficult it is to implement a safe and user-friendly multiple update system;

k A facility to limit users to a set amount of disk space (not available on all level 2 servers).

The Acorn Level 2 fileserver needs a 6502 second processor or an Acorn System 3, 4 or 5. SJ Research offer a Level 2 fileserver that will run on any CP/M system with enough RAM, plus both a Winchester-based computer purpose-built as an Econet fileserver and another which will fileserve from normal floppy drives. GSL do one to run on a BBC model B under the E-net system and Torch offer the facility for any Torch machine to read files from the disks of any other. The standard Econet can easily handle a number of fileservers. Machines do assume that the fileserver is number 254, but this number may be changed in the *I AM command (see below), by an Osword call with A=&13 or by a *FS utility if one is provided in the system library.

4.2 Fileserver commands

The commands a network machine may give to the Acorn Level 2 fileserver are given here. The other Level 2 fileservers offer a similar range of functions, usually in the same way.

a *I AM FRED *or* *I AM FRED ABC
or *I AM 247 FRED *or* *I AM 247 FRED ABC
or *I AM 3.247 FRED *or* *I AM 3.247 FRED ABC
or *I AM FRED : followed by a password on the next line, which does not appear on the screen,
or *I AM : followed by FRED and the password on the next line, neither of which appear on the screen.

where FRED is a username, ABC is FRED's password, 247 is the number

of an alternative fileserver machine and 3 is the number of another network connected via a bridge or gateway to yours. Network versions from 3.4 onwards allow the last versions of the command, with a colon in place of a password - the password is then requested on the next line and does not appear on the screen.

When the fileserver receives the *I AM command it searches the password file for a user FRED. If FRED exists it checks the password and then searches the root directory of all disks attached to the fileserver for a directory called FRED. If it does not find such a directory then FRED is allowed PUBLIC access to the root directory on drive zero; if it does, then FRED is allowed OWNER access to the directory FRED and this becomes his User Root Directory for the session. The boot option is then applied if set - it is stored in the password file and not with the directory, so it only takes effect when the owner logs in and not if anyone else gains PUBLIC access to the directory with the *DIR command. The SJ fileserver allows user root directories to be further down the hierarchy than the system root. The *I AM command is a fileserver command that is largely processed in the NFS.

b LOAD "filename" or SAVE "filename" or CHAIN "filename" for Basic programs. These commands are Basic commands that are converted by Basic into the OS *LOAD and *SAVE commands, which the NFS passes straight through to the fileserver. The fileserver will provide the file if it is in the CSD: the SJ fileserver also searches the CSL.

c *LOAD filename (start) and *SAVE filename start end (execute) (reload) to load and save areas of memory. 'Start', 'end' and 'execute' are memory locations in hexadecimal; those in brackets are optional. As on the disk system, 'end' may be replaced by '+length' if desired. Quotes are not necessary in * commands, but variables (e.g. A$, adr%) may not be used for filenames or addresses. If variables are wanted, they can be used if the * command is sent to the operating system using the OSCLI command of Basic 2.

d *RUN filename or */ filename searches the CSD (see **i**) and then the CSL (see **p**) for a file of name "filename", loads it into your machine and runs it. *filename by itself has just the same effect so long as "filename" is not a command provided in the MOS or any sideways ROMs. It is only necessary to use the *RUN command (or the abbreviation, */, and if you wonder about */ see **n**) if you wish to avoid a clash with a command provided in a sideways ROM - for example, if you wish to use the network version of *DUMP. *DUMP is a utility provided in the DFS ROM

to inspect the contents of any file. The problem is it reads information from the file byte by byte, which is slow on the net. A better *DUMP can be provided on a network fileserver, but if a station containing a DFS types '*DUMP XYZ' then the *DUMP used will be the one found in the DFS ROM. To use the *DUMP from the fileserver library it is necessary to type '*RUN DUMP XYZ' (or '*/ DUMP XYZ').

e *SPOOL filename and *EXEC filename are described in the User Guide and have just the same effect on the Network Filing System as on any other that supports them, though since they proceed byte by byte they are slow on the version 3 network.

f *DELETE filename deletes a file, if you have Owner access and if it is not locked. To delete a directory it is necessary both to empty it and to unlock it.

g *RENAME filename1 filename2 renames filename1 as filename2 (note for CP/M users - your REN command operates the other way). Rename can be used to move files between directories so long as the user has OWNER access to both the directories concerned. If a SYSTem privileged user gives the command

```
*RENAME FRED.PROG JIM.PROG
```

the file "PROG" will disappear from FRED's directory and appear in JIM's - so long as it was not locked. Renaming in this way cannot of course be done across disks or fileservers - it is necessary to use an LCOPY (logical copy) utility to copy the file.

h *CAT lists the currently selected directory (CSD).
*CAT $ lists the disk's root directory ($).
*CAT $.FRED lists FRED's directory if FRED is listed in the Root directory.
*LCAT can be used instead of *CAT $.LIBRARY, if LCAT is provided in the library.
'*CAT' can be abbreviated to '*.' (but *LCAT cannot be abbreviated to *L.).

i *DIR directoryname changes your Currently Selected Directory to the one specified. If this is not part of the tree below your own User Root Directory, you have only PUBLIC access to the files in it. *DIR by itself restores your CSD to the one you had when you logged on. On the Acorn fileserver there is no command to go back just one level in the hierarchy.

j *PASS "" ABC sets your password from nothing to ABC; *PASS ABC ERNE changes your password to ERNE. Note that it is necessary to know the old password to change it.

k *CDIR FRED will create a subdirectory FRED in the currently selected directory, assuming the user has owner access to it. Each subdirectory created takes up at least two sectors on the disk (half a K), so too many subdirectories are not a good idea unless you are using a Winchester Fileserver.

l *ACCESS filename LWR/WR controls OWNER access (before the /) and PUBLIC access (after the /) to the file. L, if given, would lock the file (absence would unlock it). Locking a file prevents you deleting it or saving a new copy over it. It does not prevent random access writes, which could wipe out all the data. Since other people can never SAVE into your file area or DELETE your files the Lock feature only applies to the Owner. Saving and deleting is controlled by L; the only other way of writing to a file is random-access writing which may be used to amend parts of it. This is controlled by the W feature. The R feature controls any form of reading. So

```
*ACCESS FRED LR/R
```

prevents any form of writing to a file but allows all to read it,

```
*ACCESS FRED LWR/
```

allows the owner to read or perform random access writes, and the public to do nothing.

The default ACCESS is WR/, allowing the owner to do anything to the file and others to do nothing - except catalogue the directory and thus find out that the file exists. On some fileservers a different default access can be specified for each directory.

D in an access string means that the entry is a Directory and not a normal File.

m *INFO filename gives the following information on the file.

File name	*e.g.*	FRED
Load Address	*e.g.*	FFFF1200
Execute Address	*e.g.*	FFFF8023
File Size	*e.g.*	0001BF

Access modes allowed	*e.g.*	WR/R
Date of saving	*e.g.*	12/10/83
System Internal Name	*e.g.*	1A5

File names should be alphanumeric only. Some punctuation is allowed, but it usually causes problems when transferring files to different systems. '$', '.' and ':' have special meanings, and '!' is by convention reserved for BOOT files. Full stops are used to separate directory names from files (or further directory names) and a colon at the beginning of a name indicates it is a disk name. The filing system ignores case (you can make your programs do the same by ANDing each character with &DF, see glossary), so FRED, Fred, and FrEd are identical. One convention useful when scanning directories is to save BASIC programs with names entirely in upper case, data files with names entirely in lower case and executable files (such as library utilities) with lower-case names starting with an upper-case character.

Addresses are given as 32-bit values in hexadecimal. If the first two bytes are 0000 the file will load into a second processor (if one is attached to your terminal): if FFFF (as above) it will load into the I/O processor. There is no difference if there is no second processor, but with second processors on a network it is vital to make sure that Econet utilities (such as NOTIFY) and screen dumps have load addresses such as &FFFF0E23 or &FFFF7C00 while applications software has addresses such as &00000400.

The 'System Internal name' or SIN is of use only to the system: it is the address of the first sector of the file on the disk. Since files are not necessarily stored in contiguous sectors on the disk this is not as useful as the equivalent given by Acorn's DFS in the event of a corrupt disk.

n *EX is equivalent to the disk command *INFO *.*, and gives the *INFO information for all the files in the CSD. It is a good idea to give this command from an 80-column mode. Why *EX? Well, it is shorter than *INFALL or *INFO *.*, someone remembered it from their university mainframe days and it is short for 'Examine'. If you want a more meaningful command, write your own system.

o *SDISC 'name' selects another disk for all subsequent operations – for the floppy-based Acorn fileserver one can say 'the other disk' since only two disks can be available. It is only necessary if the same directory occurs on both the disks, since the login procedure searches both disks for the appropriate directory. Files on other disks can be selected without changing disk simply by giving the disk name (preceded by a colon and

terminated by a full stop) before the filename – but the filename must be the full version, i.e. be preceded by the directory name (and if that directory is not in the root by any higher directories too). So

```
LOAD ":DISC2.$.FRED.PROGRAM"
```

will load a Basic file called "PROGRAM" from directory "FRED" on another disk called "DISC2", so long as that disk is available on the system and PROGRAM has public Read access (the $ may be omitted). The fileserver does not allow two disks with the same name.

p *LIB may be used to select a different library from the currently selected one (the CSL). The default library is the directory $.LIBRARY if it exists, or $ itself if it does not. Any directory may be selected as a library – there is nothing special about a library directory except that the entries would normally consist entirely of files which may be loaded as commands. This command would rarely be used: most users should be happy to use the command files saved for public use in $.LIBRARY and keep their own extra commands in their own directory.

q *OPT 1,1 turns on an automatic *INFO on each file subsequently accessed: *OPT 1,0 restores the default option of no such automatic *INFO. Note that the cassette system has additional *OPTs 1,2 and 1,3, which cause 'Bad Option' messages on the net though they are ignored on disk systems.

r *OPT 4 is rather special

*OPT 4,0 turns off the boot option applicable to the current user
*OPT 4,1 sets the boot option to *LOAD !BOOT
*OPT 4,2 sets the boot option to *RUN !BOOT
*OPT 4,3 sets the boot option to *EXEC !BOOT

The boot option is applied on logging in. If it is set to 1, 2 or 3 the system searches for a file !BOOT in the CSD and then *LOADs it, *RUNs it or *EXECs it. If it fails to find it the message 'File not found' is returned, which can be confusing as it immediately follows a login yet the login has in fact been successful. Boot options allow software to be automatically initiated as soon as a station logs on to the system.

On the network there is an additional BOOT facility available to all stations. There are two ways a network machine may get onto this
(i) by holding SHIFT down while BREAK is depressed and released;

(ii) on power-on or any BREAK depression if the internal keyboard link 5 is connected.

In either case the NFS ROM inside the machine will send the command *I AM BOOT to the currently selected fileserver. If user BOOT does not exist on that machine the reply will be 'User not known': if it does exist then subsequent activity will depend on the boot option for user BOOT and the contents of !BOOT in the root or in directory BOOT, so anything is possible.

s *BYE logs you out of the fileserver. No harm is done if you do not log out, though the next user of your machine may then access your files. If there are more stations on the net than the fileserver has been told to allow, you may stop another one logging in later.

There are some additional commands that only a system privileged user can give to the fileserver. Methods of becoming a system privileged user are outside the scope of this booklet, and those who have acquired such status legally will know the additional commands.

4.3 Reading and writing data files

Once logged into a directory on a Level 2 fileserver, a machine can read and write data files in exactly the same way as with a local disk, though without the 'Can't Extend' problems that occur with the Acorn DFS. This should apply whatever language or computer is in use. For example, if a 380Z or S100 machine is on the network with CP/M running locally, the CSD of whatever fileserver it is logged into is simply addressed as drive P and files can be copied to or from it using PIP, e.g.

```
A>P:PIP   B:=P:*.*
```

would copy all files from the fileserver CSD to the disk on the local drive B, after retrieving PIP itself from the fileserver. CP/M languages can address the fileserver similarly. However, we shall now address the matter from the point of view of BBC Basic. As with CP/M, there is no difference between the use of fileserver files and local disk files unless one wishes to be fast - it is the single-byte problem again. All the BBC Basic file-handling commands

OPENIN, OPENUP, OPENOUT, BPUT#, BGET#, PRINT#, INPUT#, EOF#, EXT#, PTR# and CLOSE#

work exactly as on disk, as do all the OS calls

OSFILE, OSARGS, OSBGET, OSBPUT, OSGBPB, OSFIND and OSFSC

though these have some Econet-specific extensions.

File-handling programs therefore transfer from disk to net or vice versa with no problems apart from speed, so long as no filing-system-specific commands have been used (e.g. the disk OSWORD &7F to read a sector, a *DRIVE command or a *OPT 3 - which is only relevant to cassette but which causes no errors on a disk system).

The single-byte problem is that all the normal BASIC operations, together with *SPOOL and *EXEC, proceed via a common BPUT/BGET interface with the operating system and this is very slow on Econet version 3. There are four ways round this:

(1) wait for the next version of the Econet;
(2) buy an add-on package that intercepts and buffers BPUT/BGET;
(3) modify the software so it uses *SAVE and *LOAD for all transfers; or
(4) modify the software so it uses OSGBPB for all transfers.

Option 1 depends on Acorn. Option 2 depends on Acorn or independent suppliers. Option 3 is sometimes easy, and can lead to software that is also very fast under all filing systems. Option 4 leads to software that will not work on cassette, but that will work equally well on disk or net.

4.4 Multiple access on the network

Though network file handling can operate in exactly the same way as disk, the network has multiple-access possibilities. Any number of users can have simultaneous read access to a file, though not all fileservers allow simultaneous writing. The Acorn system plays safe, even refusing read access to a user if another has a file open for update (and update access if another user is reading). The multiple read is most useful for common data files and to make programs publicly available: multiple write has to be arranged by the programmer. There are three possible approaches.

a Divide the file into a large number of subfiles, each of which may be independently edited. It is important that the editing does not proceed by a simple LOAD/SAVE mechanism. Why? Well, suppose two stations LOAD the same file and edit it concurrently. The first station

to SAVE it will lose its work as soon as the second station SAVEs its version. This summarises the problems involved in arranging for multiple updating. In this case it is important that a station retains control of a file while it is editing, by having it 'open for update' (on the Acorn fileserver 'open for read' will suffice, and will allow other stations to read the current version) all the time it is editing. Other stations will be denied editing access until the first one completes its job. It is useful to be able to inform other stations who is editing the file or subfile they wish to edit. This can be done by writing this information to another file which may be queried by any program that gets a 'File already open' error when it attempts an edit.

b Open for update, write the change and close again within a short time. This is particularly suitable for a file to which material is being appended - for example, a message file or one that logs events.

c Dedicate a machine to managing the file. All other users communicate with a program running in that machine. It is not vital that the machine be dedicated to this job day and night - just that a spare machine can be found to run the management program whenever anyone wishes to access the file concerned. The machine managing the file might start with software similar to the example program given in section 5.10.

4.5 Packet communications with the fileserver

Writing software to communicate with a fileserver - or to pretend to be a fileserver - is described in section 5.6. Here we shall describe what happens on the net when a simple command is sent to one. The network monitor program supplied by Acorn with all fileservers ('*NETMON', see section 7.1) can be used to observe these happenings. Suppose machine 5 sends an *I AM FRED command to the default fileserver, number 254. This causes eight 'frames', comprising two 'packets', to be sent on the net

a a scout frame, from 5 to port 99 on 254, saying 'Are you there?'. If there is no acknowledge then 5 will continue to send scout frames until it 'times out' (just over 15 seconds): the *NETMON screen will show a column of these;

b an acknowledge, from 254 to 5, saying 'Yes I am';

c the message frame, from 5 to port 90 on 254, saying 'I AM USER';

d an acknowledge, from 254 to 5, saying 'Message received'.

There is then a short delay while the fileserver checks its password file, then the reply packet is sent

e a scout frame, from 254 to 5, saying 'Are you there?';

f an acknowledge, from 5 to 254, saying 'Yes I am';

g an answer, from 254 to 5, which will consist of three 'context handles' plus a boot option byte if the login was successful and an error string (usually 'User not known') if not;

h an acknowledge, from 5 to 254, saying 'message received'.

And it all happens in a tenth of a second!

5 Writing Econet software

Writing Econet software is, for most of us, a matter of calling the *primitives* supplied in the Econet ROM. Those who wish to go further (and there is no need to do so for ordinary purposes) must write their own primitives, which involves programming the ADLC directly. The ADLC is described in Chapter 7 (though not in nearly enough detail to be programmed).

5.1 The primitives

The primitives are a set of machine-code routines which

a are called by the high-level software for all network operations; and
b may be called from your own Basic or Assembler routines.

The primitives do all the detailed work involved in transmitting and receiving messages: there is a lot more to this than meets the eye, especially in the BBC machine. Being in a sideways ROM they are only switched into the memory map for the occasional millisecond, yet they have to check every frame that passes along the network and avoid both interference with and interference from everything else that is happening in the BBC machine. On power-up they claim half a K of public workspace (space they allow a disk or other system to share) and half a K of private workspace, thus pushing PAGE up by one K (from &E00 to &1200 on a standard BBC machine). They also claim NMI workspace: half a page from &D00 where they put the small bit of code that is entered every time a frame passes along the net - code that checks the destination address in the header of the frame and wakes up the main NFS if this address is the same as that set on the machine's address links. They also alter the NET vector at &224 so that it can intercept the keyboard.

The primitives may be used without knowing anything of the workings of the ADLC or the electrical properties of the network. They occupy just over 1K starting at &E000 in the System 3, 4 or 5, &A000 in the Atom, &8000 in the BBC machine and a few K below the CCP in a CP/M system. There are ten primitives, the first of which is a pair requiring pre-arrangement between the two machines involved. The last eight are *immediate* operations, which take effect without any preparation on the

part of the distant machine - although it must of course have its Econet primitives operational (*enabled*). Machines can protect themselves from such inroads by setting a protection mask, which on Atoms is the bottom six bits in location &23B and on BBC machines is accessible to an OSWORD call with A=&13. A high-level call to *PROT sets all the protection bits; a call to *UNPROT unsets them. With old issue Econet ROMs this protection is not effective against machines numbered 240 or higher.

5.2 What we need to know before we can start

Here we shall describe how to write Econet software on a BBC machine. We shall use BBC Basic throughout since it is not necessary to be able to write in 6502 Assembler to use most of the primitives, and those capable of writing in Assembler can easily convert from Basic. It is necessary to

a understand hex;
b understand the use of AND to mask off parts of a byte or group of bytes;
c be aware of the X, Y and A registers in the 6502;
d understand the use of the **$**, ! and ? 'indirection operators' in BBC Basic;
e understand the use of DIM Buf%255 to reserve a block of memory; and
f be able to use OSBYTE and OSWORD calls from Basic.

We shall make the use of OSBYTE and OSWORD a little easier with two procedures and a function

```
DEF PROCosw(A%,Buf%)
X%=Buf% MOD 256: Y%=Buf% DIV 256
CALL &FFF1
ENDPROC

DEF PROCosb(A%,X%,Y%)
CALL &FFF4
ENDPROC

DEF FNosb(A%,X%,Y%)
=(USR(&FFF4) AND &FF00) DIV 256
```

PROCosw and PROCosb merely provide a compact way of calling OSWORD (which is at &FFF1) and OSBYTE (which is at &FFF4).

FNosb provides a compact way both to call OSBYTE and to collect the value in X after the call – it provides a *FX facility with a result. The Basic function USR jumps to the address given in brackets (which is the OSBYTE or *FX address) and returns with a 4-byte value containing PYXA in that order. ANDing with &FF00 masks out all but X, dividing by 256 shifts the masked-out X down to the bottom 8 bits so it makes a sensible value. For example, you could use

```
vdustatus=FNosb(117,0,0)
```

to obtain information on the latest settings of VDU 2, 5, 14, 15 and 21, or

```
switches=FNosb(255,0,255)
```

for information on the settings of the keyboard links (as written this gives 0 for switches that are on, 1 for those that are off).

5.3 Software provided in the NFS

The ten primitives are

1a: Transmit
b: Receive
2: Broadcast
3: Peek
4: Poke
5: Machine type
6: Remote Subroutine Jump
7: Remote Procedure Call
8: Remote Operating System Procedure Call
9: Halt
10: Continue

These operations are controlled by OSWORDs &10, &11 and &12, and OSBYTEs &32, &33 and &34. (Though OSWORD and OSBYTE are operating system calls these ones are not recognised by the OS so they

are passed around the sideways ROMs – the NFS intercepts and acts on them.)

In addition there are two OSWORD calls with special purposes

OSWORD &13

for reading and setting state information such as fileserver number.

OSWORD &14

for (a) sending a control block to the fileserver, (b) sending a text string to another station's keyboard buffer, and (c) causing a 'fatal error' in another machine. An OSBYTE call (&35) terminates remote control in the target machine.

5.4 Econet assistance provided by the MOS

The BBC machine was designed with the Econet firmly in mind, so it is not surprising to find much in the MOS to help the Econet. The main items are described here.

*FX 5,4 causes printer output to go to the network printer. If the printserver is machine number 62 it is also necessary to change the printserver number inside the BBC machine, which can be done with

```
DIM Buf%4
Buf%?0=3: Buf%!1=62
PROCosw(&13,Buf%)
```

or with a fileserver utility *PS 62. It is quite useful to have a *PS utility that both sets the printserver number *and* performs the *FX 5,4.

OSBYTE 201 is used by REMOTE to disable the keyboard. Actually it merely alters location &259: if this is other than zero the keyboard ceases to work.

OSBYTEs 206, 207 and 208 allow nearly all OSBYTE, OSWORD, OSRDCH and OSWRCH calls to be intercepted by the network. If locations &25E, &25F and &260 hold zero this interception is not activated.

The NET Vector at &224 and &225 is where the OSBYTE, OSWORD, OSRDCH and OSWRCH calls intercepted after calling OSBYTEs 206, 207 and/or 208 are sent. It is also informed every time an input line is completed: this needs no special activation.

Event number 8 is the Econet Event. If the event vector (&220, &221) is called with the value 8 in the accumulator it means there has been a remote user procedure call, and the number of the procedure required is in X and Y. This cannot happen unless this event has been enabled (with OSBYTE &0E).

The following workspace is set aside for the Econet

In zero page:

&90 to &9F	
&A0 to &A7	for the current NMI owner
&B0 to &BF	general filing system scratch space
&C0 to &CF	for the current filing system
&F4	holds the number of the currently selected paged ROM

In page 13:
&D00 to &D9E for the NMI routine
&D9F to &DEF holds the expanded vector set, which in the case of the Econet means it holds pointers into the Econet ROM for the seven (out of 27) operating system vectors that are concerned with file handling. These are normally called at addresses &212 to &21E. When Econet is active these addresses are changed to point into a special area in page &FF, which bounces calls back to an address held in this expanded vector set after changing the currently active ROM number to that held with the vector. The net result is that the values held between &DBA and &DCE are important for file handling and the Econet ROM number will be found in the seven locations &DBC, &DBF and so on in steps of three.

5.5 Primitive 1: Normal transmission/reception

Transmit and Receive provide all that is necessary for almost any network operation. This primitive is the only one required for all fileserving and printserving, and there may be many Econets on which no other primitive is ever used.

The following operations take place when a message is transmitted from X to Y

a X sets up a control block describing the message, and activates it.
b Y sets up a control block declaring its willingness to receive a message and reserving an area of memory for it, then activates it. Activation

deposits a copy of the control block in the NFS workspace, where all the complicated work takes place out of the way of any user software.
c Both stations poll their control blocks to find out
(X) if the message has been successfully transmitted. This will mean that it has definitely been successfully received. The converse does not apply: if the final acknowledge frame of a transmission is lost then the transmitting station will be led to believe that the operation failed when it did in fact succeed;
(Y) if a message has been received.
Both stations can be running other software while the transmission takes place. X should poll its control block at intervals and retransmit it if necessary (see p. 36). Y may also continue with other jobs before, during and after message reception, though if Y does not poll its control block it will never discover that the message was received.
d Y reads its control block back from the NFS workspace, and may discover from it details that it did not have to pre-arrange, such as the number of the station that transmitted the message and the actual length of the message. Y can then use the message.

It is all quite complicated, but very versatile. Details on the control blocks, activation and polling follow.

5.6 Transmission

'Transmit' transfers a block of memory from your machine to another machine, so long as it is prepared to receive it. To use Transmit it is first necessary to set up a Transmit Control Block (TxCB), which consists of 12 bytes. In the following list the bytes are numbered from 0 to 11 as this will make things easier when programming, though it does not seem natural at first. In a Basic program the TxCB for this operation can be dimensioned with the instruction DIM TxCB%11.

0 The 'control byte'. The top bit must be set: the rest of the byte is an optional 7-bit 'control code' which will be sent with your message. With a control code of zero this byte could be set with ?TxCB% = &80 or TxCB%?0 = 128.
1 'Destination Port' - an 8-bit code which you may use to tell the receiving machine the purpose of your message. You should not use port zero (which indicates an immediate operation), ports &90 to &9F (which are used by the fileserver) or &D0 and &D1

(which are used by the printserver), though you may of course use these if you wish to pretend to be a fileserver or printserver. You could specify port 8 with TxCB%?1=8.

2 The number of the station to which you wish to transmit. This is the local number, or the low byte of the 16-bit station number. 235 is the default printserver, 254 the default fileserver. To transmit to station 5 use TxCB%?2=5.

3 Zero, or the high byte of the station to which you wish to transmit. Stations on your own network have a high byte of zero. If your network is connected to others via bridges or gateways, you can address a message to another station that may be several networks away by giving its network number in this byte. Once your message reaches its destination network the high byte is set to zero by the last bridge or gateway through which it passes. Normally you would set TxCB%?3=0.

4 - 7 The 32-bit address of the beginning of your message in your RAM, low byte first. If your machine has only a 64K address space, bytes 6 and 7 will be zero. If you have put your message X$ into your location M% with the command $M%=X$, then you would enter this address with TxCB%!4=M%.

8 - 11 The 32-bit address of the byte following the end of your message, low byte first as before. Continuing with the example above, you could put TxCB%!8=M%+LEN(X$)+1 (or +2 if the carriage-return is to be sent).

In preparing the transmit block above we introduced the message rather as an afterthought in bytes 4–7, as a string in X$ which we poked into a memory buffer starting at M%. It would be most unwise to do this without previously setting M% to a suitable address, for example with DIM M%100 (to allow for messages of up to 100 bytes). However, we could instead send the whole block of function key assignments from one machine to another using TxCB%!4=&B00 and TxCB%!8=&C00, or the whole of a Basic program with TxCB%!4=PAGE and TxCB%!8=TOP+1.

5.7 Calling and polling Transmit

This is done by calling OSWORD with &10 in the accumulator and X and Y pointing to the TxCB. Using our procedure above it is one instruction

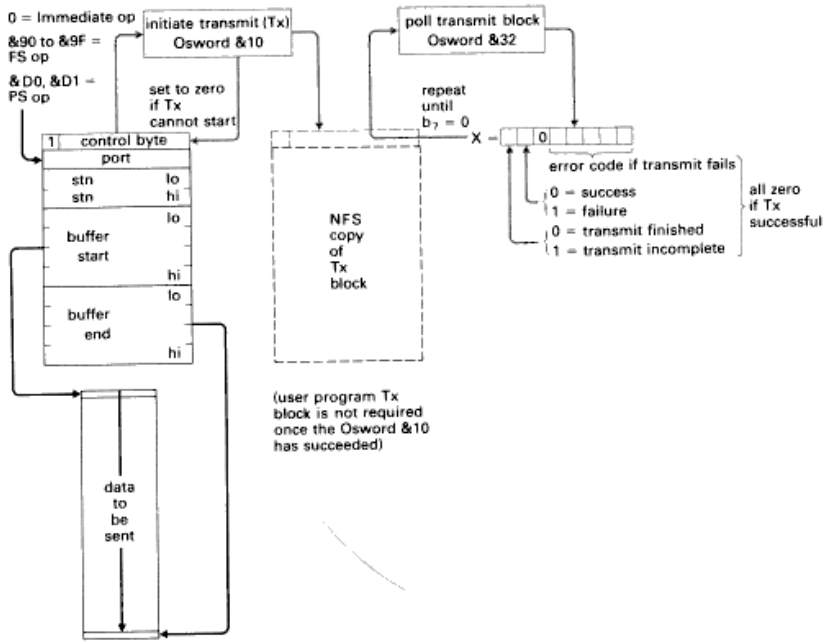

Figure 5.1 Transmit. [The grey area is common to Figures 5.1 and 5.3–5.6.]

```
PROCosw(&10,Txcb%)
```

However, if this is called while a previous transmit is incomplete, the primitives will be unable to accept the TxCB and will signify this by setting the control byte to zero. It is therefore necessary to check this immediately after the PROCosw above with

```
IF ?TxCB%=0 THEN GOTO transmit-cannot-start
```

If ?TxCB% is unchanged, then the transmit has started and should complete within a fraction of a second. It can be monitored with OSBYTE &32, which returns in X a copy of the control byte of the NFS's copy of the control block. There are no input parameters to this call as there can

never be more than one TxCB open at one time. The TxCB can be polled thus:

```
100 REPEAT X%=FNosb(&32,0,0):UNTIL X%<&80
110 IF X%<>0 THEN PRINT "Transmit error ";X% AND 31:GOTO recovery-routine
120 REM Successful transmit completed
```

Note that transmit errors are not necessarily a cause for concern. Error numbers 65 and 66 (&41 and &42, but printed as 1 or 2) are the most common (see p. 44), and the transmit should be restarted with another OSWORD &10. When programming at this level you must provide your own collision arbitration and timeout: when you use higher level facilities such as OSWORD &14 (p. 54) the NFS will retry and timeout for you. The problem with the latter arrangement is that you need to provide an ON ERROR recovery routine if you wish your program to continue.

5.8 Reception

A message cannot be received merely by calling a subroutine. Instead, a machine must declare its willingness to accept messages by activating, in the Econet workspace, a Receive Control Block (RxCB). Or two. Or three or four. Since there may be more than one RxCB they are stored in a RxCB vector or RxCBV, and on the BBC machine the NFS has room for up to 16 Rx blocks in its RxCBV.

A Receive Control Block is basically the same as a TxCB - only the meanings are slightly different and it needs to be preceded by a 'result byte', which makes it 13 bytes. It can be dimensioned with DIM RxCB%12 and prepared thus

0 The 'result byte' must start at zero, with ?RxCB%=0.

1 The 'control' or 'flag' byte must have the bottom 7 bits set to 1 and top bit set to 0 (RxCB%?1=&7F - &7F is 01111111). When a message has been received by this block this byte will be the same as the first byte of the TxCB that sent the message - i.e. 128 plus the control code, so the top bit will certainly be set.

2 The port number should be zero if messages addressed to any port are acceptable. If non-zero, the block will only accept messages sent by a TxCB with a matching port number. We could therefore use RxCB%?2=0 or RxCB%?2=8, to match the TxCB on p. 34.

3 The station number of the station from which messages are to be accepted. As the port number, this may be zero to allow reception from any station. If non-zero, messages will only be accepted from the specified station. We could thus use RxCB%?3=0 or RxCB%?3=6 (if the TxCB on p. 34 was in station 6).

4 The high byte of the station number, always zero unless you are writing code for a bridge or a gateway (RxCB%?4=0).

5 - 8 Start of the memory buffer reserved for messages that may be received by this block. This would normally have been dimensioned earlier in the program (e.g. with DIM Rbuf%255), in which case RxCB%!5=Rbuf% will do. Anywhere will do, however, and following the other two examples on p. 34 we could use RxCB%!5=&B00 or RxCB%!5=PAGE.

9 - 12 End of the memory buffer for messages that may be received by this block. As with the TxCB, this should point to the next free byte - i.e. one after the actual end of the buffer. Messages longer than this buffer will not be received (actually they will be loaded into the buffer until it is full, but the RxCB will then reset itself for another message and the user will not be informed that any reception has occurred). Three possibilities for this would be RxCB%!9=Rbuf%+256, RxCB%!9=&C00 or RxCB%!9=HIMEM.

5.9 Calling and polling Receive

Having prepared the RxCB it must be activated (*opened*) by OSWORD &11, which copies it into NFS workspace and returns in the *result byte* a handle for it, which we may call the 'Control Block Number' or cbn%. If it cannot be opened the result returned is zero (which is how the result byte starts). The following instructions are therefore necessary

```
200 PROCosw(&11,RxCB%)
210 cbn%=?RxCB%
220 If cbn%=0 THEN deal-with-the-problem
230 REM Continue with other jobs: poll the control block occasionally.
```

Polling the Rx block is done by calling OSBYTE with &33 in the accumulator and the control block number in X

```
300 REPEAT
(any other operations here)
510 PollRxcb%=FNosb(&33,cbn%,0)
520 UNTIL PollRxcb% <> &7F
```

though 520 could be UNTIL PollRxcb% > &7F if desired.

When the poll succeeds, the NFS's copy of the Rx block needs to be read back into memory accessible to our software, and this is done by the same call that opened the Rx block but with the control block number in the 'result byte' instead of zero. The contents of the next 12 bytes are immaterial because they will be overwritten by the control block read back from the NFS. It is normally convenient to re-use the original block, so reading back will simply be a matter of

```
600 ?RxCB%=cbn%          if it has not remained intact
610 PROCosw(&11,RxCB%)
```

The changes in the RxCB are now of interest. The first four bytes will now be identical with those of the TxCB in the machine that sent the message, so

RxCB%?1

will contain 128 plus the 7-bit control code of the Tx block

RxCB%?2, 3 and 4

(the port and station numbers) will have been changed only if they started at zero.

The next four, the buffer start, should not have been changed at all.

The next four, buffer end, will not have been changed if the message was exactly the expected length, but if it was shorter will have been reduced to point to the actual end of the message. As with the TxCB, the pointer is to the byte following the last byte of the message. The message can now be retrieved from the buffer, if desired.

Reading the control block deletes it. If a message is never received, another way of deleting a control block is OSBYTE call &34 with X holding the control block number

```
PROCosb(&34,cbn%,0)
```


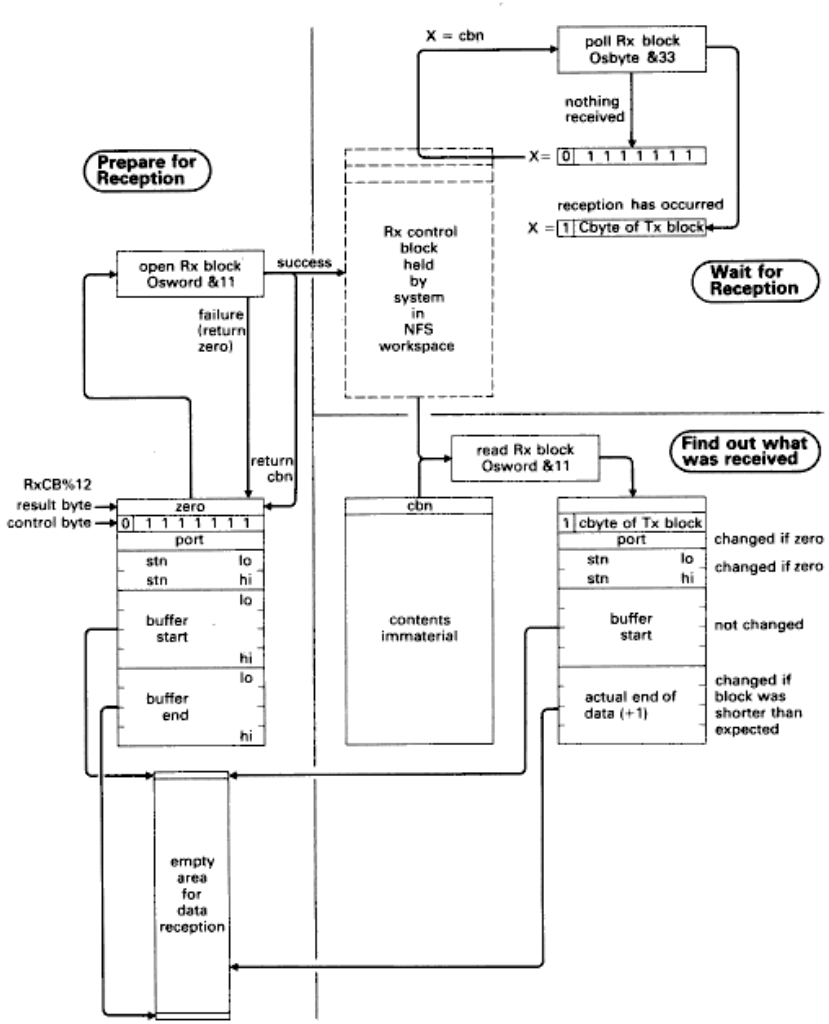

Figure 5.2 Receive

OSBYTE &34 is necessary because opening a receive block a second time will open a new one unless the previous one has been deleted by reception, by this call or by BREAK. Receive blocks are not deleted by Escape or by RUN. Programs that open receive blocks often crash mysteriously when the NFS's limit of sixteen is reached.

5.10 The Example Server Program

The program given in this section is a skeleton which allows a BBC machine to act as a server on a network. A teletext server is available which uses this skeleton, and it can easily be adapted to operate as a meteorological server or as the nucleus for a multi-user game. It is not necessarily the best way to operate such a service since it requires machines to log into it, so if they wish also to access a fileserver they need to keep changing their fileserver numbers and context handles. It does however require no software in the clients apart from that in the Econet ROM, and it does illustrate the use of many Econet functions.

A meteorological server would have temperature, pressure, wind speed and direction and other sensors attached to it – probably on the analogue and user ports. It would accept * commands requesting the values from these ports, and answer them, perhaps with the procedures used in lines 260 to 290.

The program

```
10 REM  General server, CCH Dawkins,  May 1984
20 MODE 7: @%=3: Mxst%=30: Nst%=0: Poll%=0
30 DIM St%(Mxst%),Time%(Mxst%),Rxcb%12,Rbuf%255,Txcb%15,Tbuf%255
40 FOR I%=1 TO Mxst%: St%(I%)=0: NEXT I%
```

In *line 30* of the program

St%

holds the station numbers of up to 30 logged-in stations.

Time%

holds the time at which they logged in (set in line 3470 and never used).

The receive and transmit control blocks do need to be at least 12 and 15 bytes long: the receive and transmit buffers do not for this program need to be nearly as long as they are dimensioned. In some other application they may need to be longer: there is nothing particularly magic about 255 bytes as far as Econet buffers are concerned.


```
 50
 60   LF$=STRING$(22,CHR$(10))
 70   Handle$=STRING$(4,CHR$(0)) :End$=CHR$(13)+CHR$(128)
 80   Monopoly$="Do not pass GO. Do not collect £200"
100   PROCremotecode
110   PROCSettime
120   PROCSetscreen
```

In *line 70*, three context handles are sent back to a station when it logs into a fileserver. They refer to

the User Root Directory,
the Currently Selected Directory, and
the Currently Selected Library

A fourth byte is sent from the fileserver to the client when it logs on: this tells the client whether or not to LOAD, RUN or EXEC a file !BOOT. If this fourth byte is zero the client does nothing: if it is 1, 2 or 3 the client then asks the fileserver for the file !BOOT.

Every time a station communicates with a fileserver it returns the context handles to it. For our server they are irrelevant, but the fourth byte is important because the bottom four bits of it specify the boot option and we do not wish to be asked for a file !BOOT. If the fourth byte of Handle$ is (for example) set to 2 then we must provide such a file using the Fileserver interface described in Acorn's Econet System User Guide.

In *line 100*, PROCremotecode assembles in the server machine the short bit of code we shall wish to poke across to any client who asks to 'SEE' the screen of the server – we shall return to this in PROCsendscreen.

The main loop of the program runs from line 140 to line 320. *Line 150* sets up one receive control block to receive from any station on port &99. Port &99 is the command port on the fileserver: we are pretending to be a fileserver. Though we may handle thirty (or sixty) machines, we cannot process more than one request at once so need only maintain one open receive block. If we can process each request in less than a couple of seconds we can still deal with several 'simultaneous' requests since Econet machines wait just over 15 seconds before timing out with 'Not listening'.

Lines 170 to 190 monitor the local (the server's) keyboard for operator commands, and poll the receive block set up in line 150. PROCkeymon (line 3000) only responds to the F and S keys in this program.

```
130
140 REPEAT
150   Rbn%=FNrxcb(0):REM        Set up receive block
160
170   REPEAT
180       PROCkeymon
190       UNTIL FNosb(&33,Rbn%,0)<>&7F
200
210   IF FNrxcb(Rbn%)=0 THEN PRINT "Bad Rxcb":STOP
220   Stn%=Rxcb%!3 AND &FFFF  :Rport%=Rbuf%?0
230   Bufend%=Rxcb%!9:?Bufend%=13:Message$=$(Rbuf%+5)
240   PRINT Stn%;TAB(6);Message$
250
260   IF Message$="TIME" THEN PROCreply(4,0,FNgettime(TIME)+End$)
                                                        :GOTO 320
270   IF LEFT$(Message$,4)="I AM" THEN PROClogin
                                                        :GOTO 320
280   IF Message$="SEE"  THEN PROCsendscreen:PROCreply(4,0,LF$+End$)
                                                        :GOTO 320
290   IF Message$="GO TO JAIL" THEN PROCreply(4,0,Monopoly$+End$)
                                                        :GOTO 320
300   PROCreply(0,&FE,"Unrecognised command")
310
320   UNTIL 0
330 END
```

Line 210 reads the receive block from the Econet workspace back into our own Rxcb: if the control byte is zero there is a serious fault in the network software.

Lines 220 and 230 read the transmitting station number, the port on which it awaits our reply and the message.

Lines 260 to 300 are where the meat of the serving software will reside. There are two important points. First, there must always be a *reply*. In PROCsendscreen we do lots of other things to the client, but whatever we do (within reason) it still awaits a reply on the reply port it gave us. I have used this here to send 22 linefeeds. Second, the command and reply codes used in PROCreply need to be chosen with care. They are further discussed under 'PROCreply' below.

Function Rxcb (*Zero or Block number*) deals both with setting up the RxCB (line 150) and reading it back (line 210), since both involve the same OSWORD call. The only differences are in the first byte: its contents before the call and its meaning afterwards (see Figure 5.2 and section 5.9). *Line 1020* is a shorthand way of getting four bytes in at once: &7F into Rxcb%?1, &99 into Rxcb%?2 and zero into the next two bytes.


```
1000 DEF FNrxcb(Rbn%)
1010   Rxcb%?0=Rbn%
1020   Rxcb%!1=&0000997F :REM Receive from any station on port &99
1030   Rxcb%!5=Rbuf%
1040   Rxcb%!9=Rbuf%+255
1050   PROCosw(&11,Rxcb%)
1060   =?Rxcb%
```

PROCreply (*command code*, *return code*, *string*) depends on the fact that the Econet software in the client will have a receive block open waiting for a transmit from our server on port 'Rport'. Two important bytes are sent with the reply - the command code and the return code.

The *command code* is zero if the operation is complete, 4 if the reply is an INFO message and 5 if the reply is to a successful login. Other values should be avoided without referring to the Acorn's Econet System User Guide.

The *return code* is zero if there is no error, otherwise it is an error number and the rest of the block should be an error message terminated by CR (&0D). This termination is arranged by the poke in line 2080.

```
2000 DEF PROCreply(cc%,rc%,String$)
2005   Fe%=FALSE
2010   REPEAT Txcb%?0=128
2020          Txcb%?1=Rport%
2030          Txcb%!2=Stn%
2040          Txcb%!4=Tbuf%
2050          Txcb%!8=Tbuf%+3+LEN(String$)
2060          Tbuf%?0=cc%
2070          Tbuf%?1=rc%
2080          $(Tbuf%+2)=String$
2090          PROCosw(&10,Txcb%):     REM Call transmit
2094   UNTIL ?Txcb%=0 OR FNpolltx
2095   IF ?Txcb%=0 THEN PROCtxerror(1)
2096   IF Fe% THEN PRINT "Fatal Error in PROCreply"
2098   ENDPROC
```

FNpolltx is used to poll the transmit blocks set up by PROCreply, PROCpoke and PROCcall, in case of transmit failures. It should be read in conjunction with Figure 5.1.

```
2100 DEF FNpolltx
2110   Poll%=Poll%+1
2120   REPEAT
2130       X%=FNosb(&32,0,0)
2140       UNTIL X%<128
2150   IF X%=0 THEN Poll%=0: =-1
2160   PROCtxerror(3)
2170   IF Poll%<50 THEN Poll%=Poll%+1: =0
2180   PROCtxerror(2): Poll%=0: Fe%=TRUE: =-1
```

PROCtxerror (error number) could be improved in a number of ways depending on the user's requirements. The error indications given by X% could be spelled out

X%=64 Line jammed
X%=65 No reply or collision
X%=66 Not listening
X%=67 No clock
X%=68 Bad transmit block in this machine

```
2200 DEF PROCtxerror(e%)
2210   PRINT "Error in transmit to station ";Stn%;
2220   IF e%=1 THEN PRINT " previous Tx continuing; retrying"
2230   IF e%=2 THEN PRINT " taking a very long time"
2240   IF e%=3 THEN PRINT " error number ";X%
2250   ENDPROC
```

PROC osword, PROC osbyte and Function osbyte are as in section 5.2.

```
2300 DEF PROCosw(Acc%,Buf%)
2310   A%=Acc%: X%=Buf% MOD 256: Y%=Buf% DIV 256
2320   CALL &FFF1
2330   ENDPROC
2400 DEF PROCosb(Acc%,x%,y%)
2410   A%=Acc%: X%=x%: Y%=y%: CALL &FFF4
2420   ENDPROC
2500 DEF FNosb(Acc%,x%,y%)
2510   A%=Acc%: X%=x%: Y%=y%
2520   =(USR(&FFF4) AND &FF00) DIV 256
```

The remotecode, when poked across and called, performs the following functions in the client

a It retrieves the arguments sent with the call. As no arguments are sent by this program, it is safe to put them at location &84, which is immediately after the end of this code.
b It switches the screen to MODE 7, without changing HIMEM. It is then safe to poke the server's screen across, so long as this has not been scrolled by hardware. This condition is satisfied by line 3230 in this program.


```
2600 DEF PROCremotecode
2610   P%=&70
2620   Oswrch=&FFEE:Osword=&FFF1
2630   [OPT 1
2640    LDX #&84: LDY #0: LDA #&12: JSR Osword
2650    LDA #22: JSR Oswrch
2660    LDA #7:  JSR Oswrch
2670    RTS
2680   ]
2690   ENDPROC
```

PROCsendscreen pokes the remotecode across to the client, calls it and then pokes the screen.

```
2700 DEF PROCsendscreen
2710   PROCpoke(Stn%,&70,&14,&70)      : REM Poke remotecode
2720   PROCcall(Stn%,&70)              : REM Call it
2730   PROCpoke(Stn%,&7C00,&3C0,&7C00): REM Poke screen
2740   ENDPROC
```

PROCpoke is a general purpose routine with the following parameters

mc%: The number of the machine we wish to poke.
adr%: The start address in our machine.
Len%: The length of the block we wish to poke.
remadr%: Where we wish the block to be poked in the remote machine.

```
2800 DEF PROCpoke(mc%,adr%,len%,remadr%)
2805   Fe%=FALSE
2810   REPEAT Txcb%?0=&82:Txcb%?1=0
2820          Txcb%!2=mc%
2830          Txcb%!4=adr%       +&FFFF0000
2840          Txcb%!8=adr%+len% +&FFFF0000
2850          Txcb%!12=remadr%  +&FFFF0000
2860          PROCosw(&10,Txcb%)
2870   UNTIL Txcb%?0=0 OR FNpolltx
2880   IF Txcb%?0=0 THEN PROCtxerror(1)
2885   IF Fe% THEN PRINT "Fatal error in Poke"
2890   ENDPROC
```

To follow the details refer to Figure 5.3, page 48

PROCcall enables a call to be made to any location in any other machine on the network, the parameters being simply the machine number and the call address.

```
2900 DEF PROCcall(mc%,remadr%)
2905   Fe%=FALSE
2910   REPEAT Txcb%?0=&83:Txcb%?1=0
2920          Txcb%!2=mc%
2930          Txcb%!4=0
2940          Txcb%!8=1
2950          Txcb%!12=remadr%  +&FFFF0000
2960          PROCosw(&10,Txcb%)
2970   UNTIL Txcb%?0=0 OR FNpolltx
2980   IF Txcb%?0=0 THEN PROCtxerror(1)
2985   IF Fe% THEN PRINT "Fatal error in Call"
2990   ENDPROC
```

The remaining procedures in the program are straightforward for anyone who has reached this point.

```
3000 DEF PROCkeymon
3010   IF INKEY(-68) THEN PRINT "Free space = ";
                                  (!4 AND 65535)-(!2 AND 65535)
3020   IF INKEY(-82) THEN PROCSetscreen
3050   ENDPROC
3100 DEF PROCSettime
3110   INPUT "Give time in hours,minutes,seconds:" hr%,min%,sec%
3120   TIME=((hr%*60 + min%)*60 + sec%) *100
3130   ENDPROC
3199
3200 DEF PROCSetscreen
3210   X$=CHR$(141)+"Screen of server machine"
3220   PRINT '' X$ ' X$ ''
3230   VDU 28,0,24,39,10
3240   ENDPROC
3299
3300 DEF FNgettime(T%)
3310   Hr$=STR$((T% DIV 360000) MOD 24)
3320   Mn$=STR$((T% DIV   6000) MOD 60)
3330   Sc$=STR$((T% DIV    100) MOD 60)
3340   =Hr$+":"+Mn$+":"+Sc$
3399
3400 DEF PROClogin :LOCAL S
3410   IF Nst%=Mxst% THEN PROCreply(0,&B8,"Too many users"):ENDPROC
3420   S=0
3430   REPEAT S=S+1
3440     UNTIL St%(S)=Stn% OR St%(S)=0
3450   IF St%(S)=Stn% THEN PROCreply(0,&D0,"Already logged in")
                                                        :ENDPROC
```



```
3460   Nst%=S
3470   St%(S)=Stn%:Time%(S)=TIME
3480   PROCreply(5,0,Handle$)
3490   ENDPROC
```

5.11 Primitive 2: Broadcast

The Broadcast operation is a special case of Transmit/Receive: it differs in the following respects

a It is limited to an 8-byte block, because it is actually transmitted as a scout packet and is not acknowledged. The lack of acknowledgment is necessary (i) because the transmitting station cannot be expected to know how many stations to expect acknowledgements from and (ii) because the acknowledgements would all attempt to transmit simultaneously and dozens (hundreds, thousands?) of machines doing this would severely test the collision arbitration facilities.

b The station numbers for both transmit and receive blocks should be &FF, though zero will be quite acceptable on the receive block since this indicates a readiness to receive from any other station.

c On NFS's before 3.6 the eight data bytes arrived in the locations reserved for the buffer start and end pointers (RxCB%?5 to RxCB%?12). On NFS's from 3.6 onwards the data bytes arrive in the usual place, the receive buffer, and the receive control block contains the pointers to this.

5.12 Primitives 3–10: Immediate operations

(3) PEEK is an immediate primitive, but it does no damage to the distant machine (apart from halting it if it is playing an Acornsoft game). It simply copies an area of memory from the remote machine to yours. It and the other seven are all called in the same way as TRANSMIT, by OSWORD &10, with X and Y pointing to the control block and subsequent polling if desired with Osbyte &32. The control block for PEEK is 16 bytes, so may be dimensioned with DIM IoCB%15

0 The control byte: &81 (decimal 129)
1 The port: zero for all immediate operations

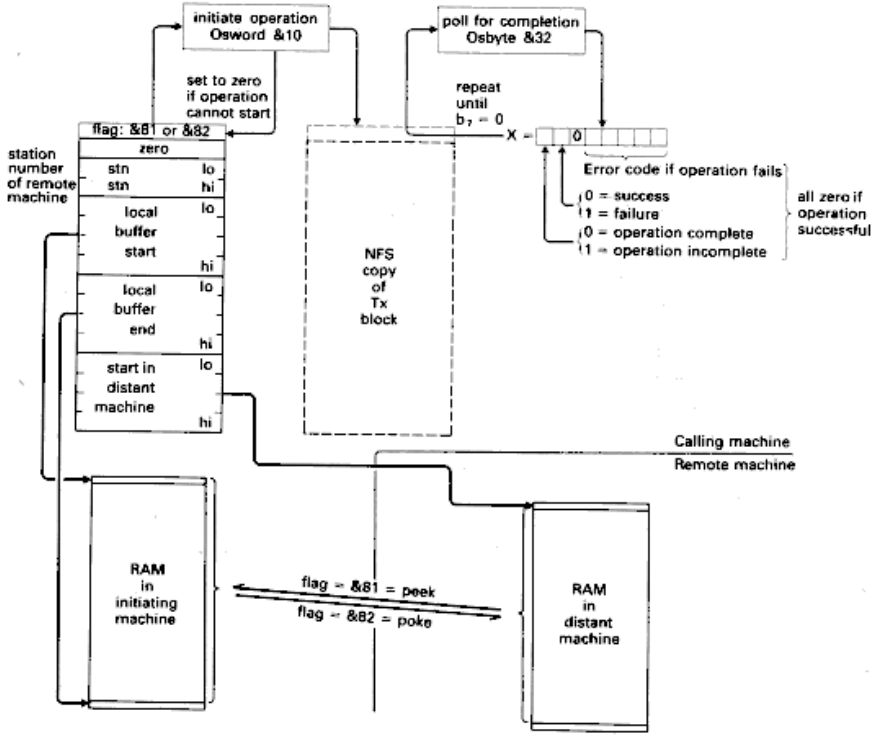

Figure 5.3 Immediate operation control blocks: (1) Peek and Poke

2,3 Station number (so 3 will normally be zero)
4 - 7 Start memory location in your machine (for reception of what is PEEKed)
8 - 11 End memory location in your machine
12 - 15 Start PEEK location in distant machine

(4) POKE is the exact opposite of PEEK. It has the same control block except that the control byte is &82, and it moves the data in the opposite direction.

(5) MACHINE TYPE: This primitive is also called a 'LOCAL PEEK', because you are peeking an area of the distant machine to tell you what sort of machine it is. In different machines this will be different areas; the area you peek is thus decided by the machine you are peeking. The low-level primitives of every machine on the network have a pointer to an area

within them that defines what sort of machine they occupy. The control block for this primitive is 12 bytes as follows

0 The control byte: &88
1 The port: zero as usual
2,3 Station number you wish to peek
4 - 7 Start of your buffer for the result
8 - 11 End of your buffer for the result

The result, in your local machine, will be (starting from the beginning of your local buffer)

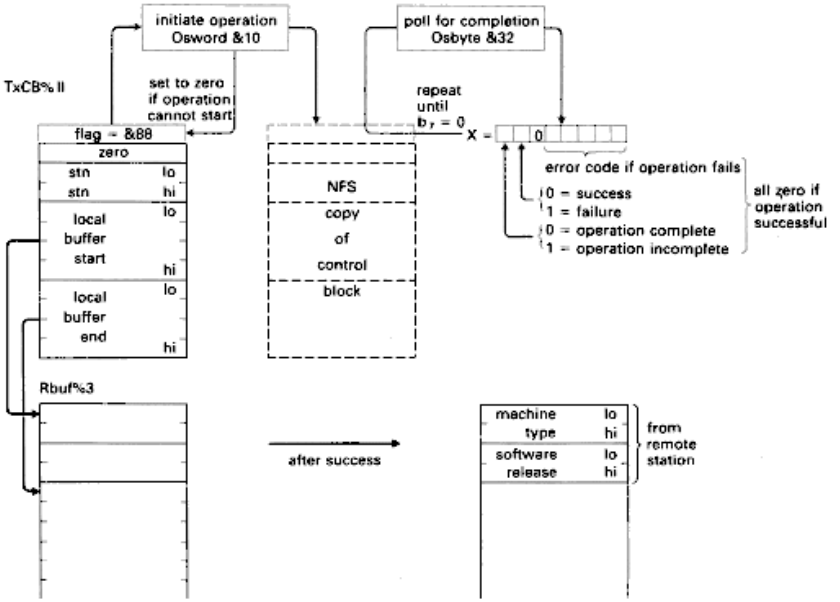

Figure 5.4 Immediate operation control blocks: (2) Machine type

bytes 0,1: Machine type (BBC machine = 1, Atom = 2, System 3 or 4 = 3, System 5 = 4, 380Z or any CP/M system = &FFFF, 480Z = &FFFD, Nascom-2 = &FFFC, SJ Fileserver = &FFFE).
bytes 2,3: Software release number in the machine we peeked.

(6) REMOTE JSR forces a subroutine jump in the distant machine to an address specified by your machine. A block of data may be transferred to

the remote machine so that the subroutine has something to work on: the address of this block in your machine must be given in the control block you use to initiate the remote JSR. You will often not need to send any arguments, in which case set the buffer start to 0 and the buffer end to 1, as the counting is exclusive. Even if you do not send any data you must still, in the remote subroutine, use the call to retrieve the arguments you did not send, otherwise you will not be able to do another remote JSR on that machine. The call, as before, is the same as for TRANSMIT, and the control block is

0	&83
1	Zero
2,3	Station number of the remote machine
4 - 7	Address of the start of your argument block
8 - 11	Address one beyond the end of your argument block
12 - 15	JSR address in the remote machine

When the remote subroutine is entered the protection bits in the machine are set to prevent a re-entry if another call to it arrives. They are only reset when the argument buffer is read. This is done with OSWORD &12, pointing to a buffer large enough for the arguments. There is therefore a need (and provision!) for a call to find out the size of the buffer, and this is OSWORD &13 with reason code 9 (see p. 52). If no arguments have been sent a buffer of a few bytes is all that is needed (the calling station number is always provided at the beginning of the arguments): if a few bytes from &70 are free then all that is necessary in the remote routine is

```
LDX #&70: LDY #0: LDA #&12: JSR &FFF1
```

(7) REMOTE PROCEDURE CALL forces a subroutine jump in the remote machine to an address specified in the remote machine. The control block is the same as for the REMOTE JSR except the control byte is &84 and the start address (12-15) is the procedure number, of which only the low 16 bits are relevant. When the call happens in the remote machine an EVENT occurs, with 8 in the accumulator and the procedure number supplied in X and Y. It is up to the software in the remote machine to process this as required. As with the remote subroutine jump, it is important to read the arguments passed even if there are none.

(8) REMOTE OS PROCEDURE CALL is the same as the REMOTE

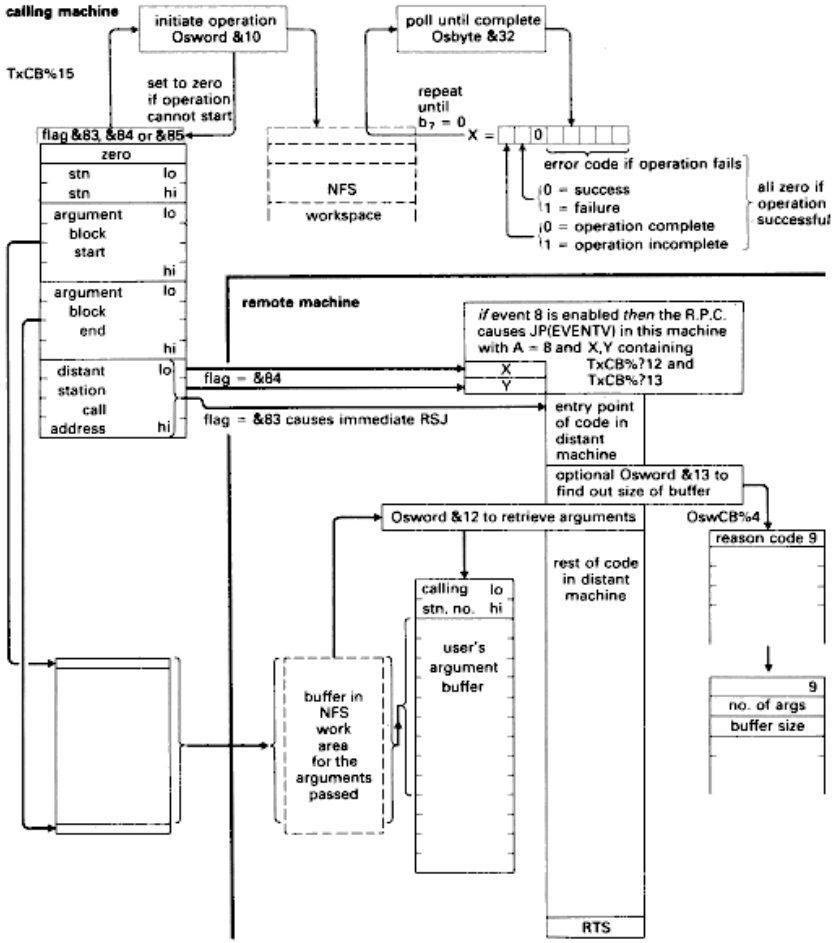

Figure 5.5 Immediate operation control blocks: (3) Remote subroutine jump, Remote procedure calls

PROCEDURE CALL except the control byte is &85 and it is not available to 'ordinary users' because control is passed straight to the MOS.

(9) HALT causes all user processes in the halted machine to cease: interrupts remain enabled and operating system routines continue to function. The control block needs only four bytes

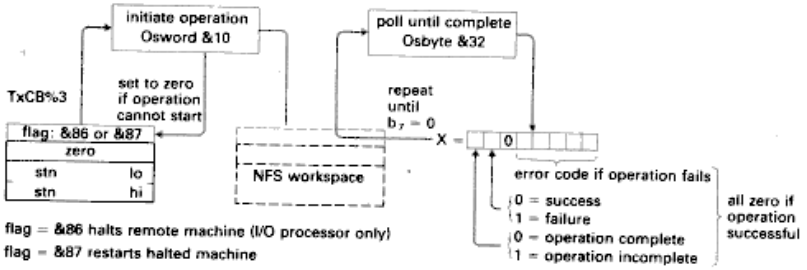

Figure 5.6 Immediate operation control blocks: (4) Halt and Continue

0 Control byte, &86
1 Port, zero
2,3 Station number of machine to be halted

(10) CONTINUE cancels HALT, and is the same in all respects except that the control byte is &87.

All these immediate operations are called in the same way as TRANSMIT (Figure 5.1).

5.13 Reading and setting state information: OSWORD &13

If a suitable control block is dimensioned with DIM OswCB%3, and a 'reason code' poked into the beginning of this block with ?OswCB% = *n*, the instruction

```
PROCosw(&13,OswCB%)
```

may be used to obtain the following information

n = 0 read current fileserver number (two bytes)
n = 2 read current printserver number (two bytes)
n = 4 read protection byte
n = 6 read context handles (three bytes)
n = 8 read station number (from your own ID links - two bytes)
n = 9 read (at the beginning of a subroutine called with a remote subroutine jump) number of arguments and buffer size (one byte each)
n = 10 read error number

The results are returned in OswCB%?1 to OswCB%?3, depending on the number of bytes.

'Read error number' is necessary because the only error numbers that may be supplied from a filing system to a BBC machine are A8 to C0 (168 to 192), but the Acorn fileserver has around 100 in its repertoire. So A8 (168) is a 'composite' number, and when it is sent the actual number needs to be obtained with this Osword call. However, such problems need not concern the ordinary programmer as all the common messages are explained with error strings - it will only puzzle people who are in the habit of retrieving error numbers with PRINT ERR and find that most fileserver errors are error number 168.

The observant reader will have spotted four reason codes omitted from the previous list. These are the corresponding 'set' operations and require OswCB%?0 to contain the reason code and OswCB%?1 onwards to contain data for the operation.

n = 1 set fileserver number
n = 3 set printserver number
n = 5 set protection mask
n = 7 set context handles

The protection mask is a single byte in which the bits have the following effects when set

Bit 0: Peek not allowed
1: Poke not allowed
2: Remote JSR not allowed
3: Remote Procedure Call not allowed
4: Remote OS Procedure Call not allowed
5: Halt not allowed

The Machine Type peek and Continue cannot be protected against, but these cause no damage and reveal no sensitive information.

It is possible to be logged into several fileservers at once. When changing from one to another, it is necessary (if one wishes to return to each in exactly the same context as on leaving it) to preserve the fileserver number and context handles somewhere else before logging into another: on restoring them it is not necessary to log back in. It is only necessary to preserve and restore those five bytes. That is the purpose of reason codes 0,1,6 and 7.

5.14 Miscellaneous applications: OSWORD &14

OSWORD &14 has three quite different functions. It can be called with

```
PROCosw(&14,OswCB%)
```

and, as with OSWORD &13, it is the first byte (called the 'flag byte') in OswCB% which determines the function.

Flag byte zero: Fileserver call

For these operations OswCB% needs to be DIMmed to at least 50.

OswCB%?1	should contain the size of the block (which will be at least 7)
OswCB%?2	should be zero (the NFS will put a Reply Port in here when it sends the block out)
OswCB%?3	is the Function Code - effectively a command to the fileserver
OswCB%?4	should be zero
OswCB%?5	should be zero
OswCB%?6	should be zero
OswCB%?7	onwards holds any other information to be sent to the fileserver

The NFS will fill in byte 2 with a reply port and bytes 4 to 6 with the current context handles as it sends the block to the fileserver, but the user has no control over this. The fileserver's reply will start in OswCB%?2 with a *command code* (zero if the operation is complete). OswCB%?3 will contain a *return code*, which will be zero if there were no errors, otherwise an error code followed ($(OswCB% + 4)) by an error string.

The function codes are given briefly here: many of the operations can be performed more easily in another way and those who need to use these function codes should refer to the Acorn Econet System User Guide for further details.

Code	Meaning
0	Decode and obey command line, which is $(OswCB% + 7)

1	Save a file
2	Load a file
3	Catalogue /*EX a directory
4	Obtain catalogue header for CSD
5	Load a command file
6	Open a file
7	Close a file
8	Get a Byte
9	Put a Byte
10	Get a block of bytes
11	Put a block of bytes
12	Read file pointer, extent and size
13	Set file pointer
14	Read names of fileserver disks
15	Read machine numbers and user ids of logged on users
16	Read date and time
17	Find out if at End-of-File
18	Read the attributes of a file
19	Set the attributes of a file
20	Delete a file
21	Read names of current disk, directory and library
22	Set a boot option
23	Log off
24	Read any named user's information (machine number and privilege status)
25	Read fileserver version number
26	Read free space on fileserver disk
27	Create directory, specifying size
28	Set fileserver time and date

Example: Obtaining date and time from the fileserver.

```
700 DIM OswCB%50
710 OswCB%?0=0
720 OswCB%?1=8
730 OswCB%?2=0
740 OswCB%?3=16
750 OswCB%!4=0
760 PROCosw(&14,OswCB%)
770 Day% =OswCB%?4
780 Mon%=OswCB%?5 AND 15
```

```
790  Year%=81+(OswCB%?5 DIV 16)
800  Hour%=OswCB%?6: Min%=OswCB%?7: Sec%=OswCB%?8
```

Lines 710 to 740 could be replaced by !OswCB% =&10000800.

Flag byte one: string to keyboard buffer

This provides a very simple way of sending a text string to another station, where it will be put in that station's keyboard buffer. OswCB%?1 and OswCB%?2 should contain the station number (so OswCB%?2 would normally be zero), and OswCB%?3 onwards should contain the string, terminated by carriage return (&0D) or zero.

Flag byte two: cause fatal error

This, in Econet version 3, is the third and last function of OSWORD &14. The OswCB need only be three bytes long: the first being the flag (2) and the next two being a station number. The call will cause a 'fatal error' on the destination machine, putting it back in command mode so that if lines of text are then sent to its keyboard buffer using OSWORD &14 with flag byte 1 they should be processed as commands. This error may be trapped by the ON ERROR call in Basic 1, but is not trappable by Basic 2.

6 Cables, connectors and terminators

6.1 The signals on the Econet

Econet cable has to carry two signals: clock and data. It would be quite feasible to mix these and convey the signal on one pair of wires, but each machine would need more input and output circuitry (a clock for output and a phase-locked-loop for input). The standard Econet needs one pair of wires for the clock and one pair for the data. A 3-wire system could be used if clock and data shared a common earth, but this would sacrifice the advantages of a balanced line system. Balanced line? Balanced line means that the two wires of the pair always carry opposite voltages. When the signal goes positive on one, it goes equally negative on the other. The voltage about which the signals swing positive and negative is called the 'common mode' voltage and does not have to be the earth potential: in practice the Econet data lines swing around 2.5 V (being about 1 V/4 V and 4 V/1 V) while the clock lines are the same as the data on the Acorn system and about 2 V above it on the SJ system. The signal is detected by a differential receiver, which operates only on the potential difference between the pair of wires and ignores the common mode voltage, whatever it is. This provides two benefits

a the radio-frequency radiation from one cancels that from the other, so we do not infringe public authority regulations by broadcasting (it would be on the Long Wave band); and

b any interference (e.g. inductive pick-up) should affect both wires equally so will be ignored by the system - although this does not protect against nearby lightning strikes.

Electrically, the cable needs to have as low a d.c. resistance as possible to reduce the voltage drop on a long net, and a characteristic impedance of 100 Ω so as to match the 75159 line driver chips in each BBC machine. The resistance requirement means longer nets need more expensive cable, and the impedance requirement is met by few cables currently in use. The pair of wires carrying each balanced line signal needs to be a 'twisted

pair'. An earthed shield around the signal wires reduces all forms of interference.

The earth is in theory not necessary for the basic system, and earths can cause problems if there are different earth potentials in different parts of a site. During early Econet development there was disagreement over the value of the Earth and issue 1 was often used without it, but it is now favoured and is a vital part of the SJ clock/terminator system. However, there may be a need to leave it unconnected to some machines.

6.2 The cable

We therefore need a cable that contains two twisted pairs and an earth, which should form a shield around the pairs. Making connections to a shield is not always easy, so the earth is preferably a naked 'drain' wire that makes contact at intervals with the shield. There are half a dozen alternative cables on the market and the choice is not easy, although it is to be hoped that a clear winner will emerge in time.

For short networks any five-wire leads will do: the signal pairs need not be twisted. The computer-to-wall-socket leads provided by Acorn with standard issue Econet machines do not have twisted pairs (they are plain audio DIN leads, such as may be purchased from any Hi-Fi shop), and

RS flex 367-448 at £ 8.62 for 25 m (resistance 88 Ω/km)
367-347 at £ 8.29 for 25 m (resistance 314 Ω/km)
367-959 at £41.75 for 100 m (58 Ω impedance flex)
or 367-921 at £69.75 for 100 m (108 Ω impedance cable)

can all be used for networks of 50 m to 300 m - the more expensive of these for the longer networks. Networks extending to over 500 m need heavier material: BICC cable No. CS7227 at £850 per 2 km from BICC or 99p per metre from Acorn, SJ flexible cable NW-CAB at 80p per metre or Anixiter cable 2-pair 1/0.9 at £54 for 100 m. The situation with respect to cables is changing fast as the Econet market develops: the things to check are

a does it have five conductors? These may be arranged as two twisted pairs or (better) as one twisted Quad, plus an Earth.
b is it flexible or not?
c if it is for a long network, will the end-to-end resistance be less than 15 ohms?

d is its characteristic impedance around 100 ohms?
e what is its overall diameter?
f does it have four different colours for the four wires?
g is it available in a convenient reel size?

as well as the normal considerations of price and delivery.

6.3 Cable wiring

It is a good idea to have a consistent colour code within one site. Though there is no need to maintain consistency between sites we suggest the colour codes below so as to assist the travelling repair man. If cables not mentioned have similar colours they could be wired in a similar way.

Signal	SJ, or RS 367-448	RS 367-577	RS 367-921	BICC CS	Anixiter	DIN pins
Data +	Green	White	Black	White	Blue	1
Data –	Red	Red	Red	Orange	Orange	4
Earth	Screen	Screen	Shield	Shield	Screen	2
Clock –	Blue	Blue	Black	Blue	Brown	5
Clock +	Yellow	Yellow	White	White	Green	3

(On some cables two wires are the same colour. In such cases the two of the same colour are always twisted around two of different colours, so they can be identified although it is not always easy.)

6.4 Sockets

Setting up a network simply involves laying the cable along the site and putting DIN sockets on it wherever a computer may be required. RS sockets 478.633 at 85p each are the best if you provide your own boxes (standard electrical boxes with blank cover plates work well), alternatively both SJ and Acorn do a twin-socket box that is specifically designed for Econet wiring. These are wired using the 'insulation displacement' technique, which is fast and saves soldering. For large sites an alternative source of insulation displacement junctions is Krone (UK) Ltd at 0242 584900. Remember that time and trouble spent in the initial wiring of a site will pay dividends later in reliability.

All sockets on the network are connected in parallel. The ideal network

is a straight line. T-junctions will cause trouble unless special 'joint' boxes (which need power) are installed: the only T-junctions allowed are the connections to BBC machines and these should not be longer than 1 m. Longer T junctions can work, but only near the clock and often only for communications between certain points. The plugs on these connections are wired 'straight' (pin 1 to pin 1, 2 to 2 and so on), as is the rest of the network.

6.5 Junctions

Junctions, to enable sections of the network to be rapidly isolated from each other, speed up fault-finding on an Econet. Each junction needs a socket on either side of the break so that when a disconnection is made a terminator or test probe can be plugged into each side. Such a junction then provides a convenient plug-in point for a clock system with two line drivers (intended to drive the two halves of a net split in this way). The connection between the sections needs to introduce as little resistance and reflection-potential as possible, so should ideally involve gold/gold connectors (Figure 6.1) rather than 5-pin DINs (Figure 6.2). However, wiring junctions up as shown in Figure 6.2 is so much faster and less error-prone than as in Figure 6.1 that the small benefit offered by gold/gold contact can be ignored.

6.6 Termination

Signals at Econet frequencies are liable to get reflected from the ends of the cables: minor reflections could also arise from bad connections in the middle. Reflections can be prevented by proper 'termination', which in its simplest form means resistances (100 Ω) across the twisted pairs to provide a load to sink the current and thus damp the ringing that occurs. The Econet circuit on older machines had provision for a terminator circuit, but this required that the end-of-network machines were permanently sited and, for 'active termination', always on. This was a nuisance, hence the current trend for separate boxes.

Passive termination (the two resistors mentioned above) does not provide the bias on the data lines which is required for proper operation of the line receivers - without it they are liable to detect noise. This bias (Data+ needs to average about 0.6 V above Data−) can only be provided if power is available. Providing power to both ends of a network cable

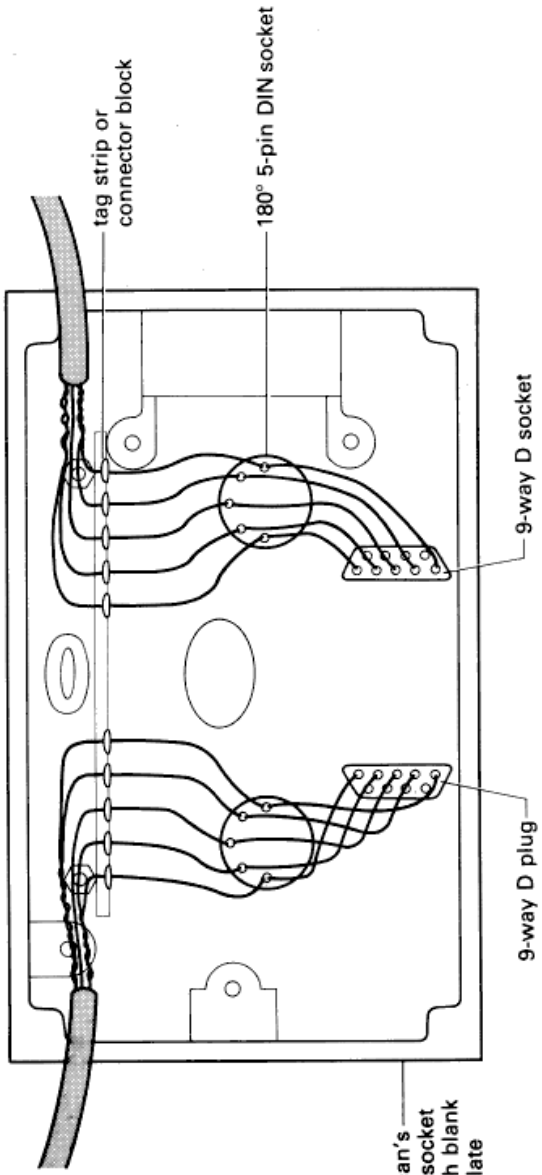

Figure 6.1 To simplify the diagram, the wires to the D plug and socket have been drawn shorter

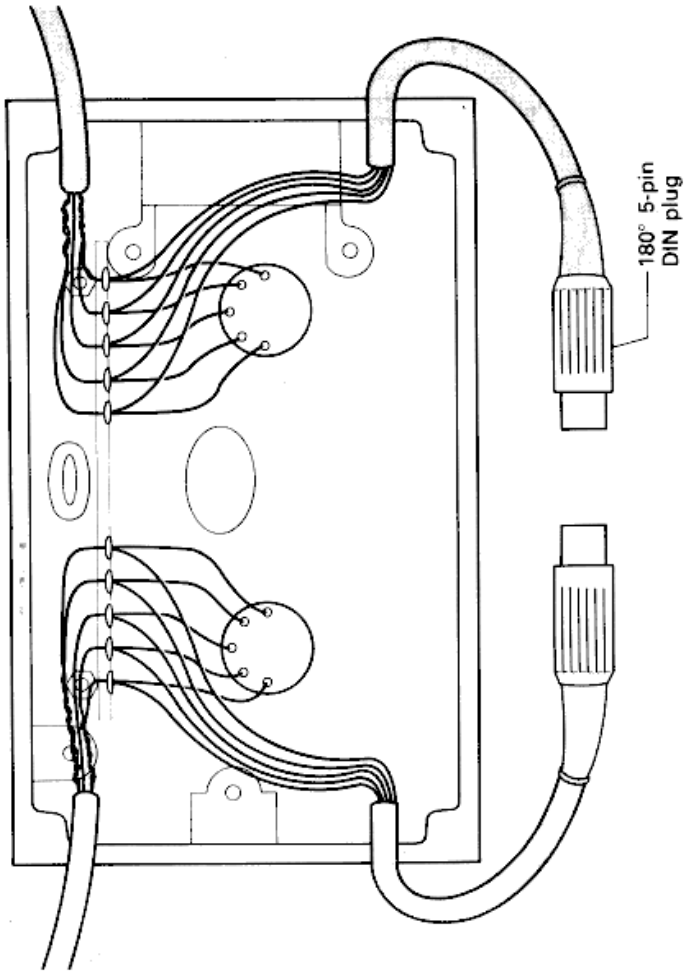

Figure 6.2 The wires to the DIN plugs would be longer in reality

could be a nuisance, so the newer systems provide the power along the cable from the clock box, and the terminators are just a few components wired into a DIN plug or a socket box. Acorn's original system required mains power at the clock and at both terminators, but it could work without a network earth.

When the network termination is inadequate (or excessive) packets are not detected properly and everything slows down as machines keep retrying. Messages such as 'Line jammed', 'Net error' or 'No reply' may appear. Chapter 9 describes how to deal with these problems.

7 The 6854 Advanced Data Link Controller

The Motorola 6854 Advanced Data Link Controller provides the link between the network and the central processor. One side of it is connected to the Econet line driver and receiver chips, the other to the data and control lines of the 6502 (as shown in Figure 8.1). The BBC version is the 68B54, the B indicating that it will run at 2 MHz. It has more registers in it than the 6502 (8, 9 or 12 depending on one's definition of a register: the 6502 has 6 or 7). Two of these share one address, one has two addresses which have different effects and one has two addresses with the same effect, so those who write the low-level Econet software need to read the 6854 manual many times! It is programmed by the 6502 to perform the Econet functions listed below - it could instead be programmed quite differently to suit the characteristics of quite different networks (though it would also need different hardware to transmit and receive on them).

7.1 Acorn's *NETMON program

One of the most useful items of software provided with an Acorn file-server is a network monitor program which, running on a BBC machine, shows the contents (up to the first 16 bytes) of every frame transmitted on the network. Interpreting its display requires - and also provides - a good knowledge of the frame structure and of some properties of the ADLC. The description that follows thus refers to the *NETMON display where appropriate.

7.2 What the ADLC does

a It monitors the communications line and - if programmed to do so - interrupts the 6502 when a frame is detected. The 6502 must then decide whether the message is addressed to it, by comparing the destination address at the head of the frame with its own address (which it reads from &FE18 on the BBC machine). In this respect the 6854 is not as advanced as the Z80 SIO (Appendix 1), which can recognise the address on its own without interrupting the central processor.

b It checks that messages received are both valid and correct. If frames do not have the expected structure they are declared invalid and bit 1 of status register 2 (location &FEA1) is set to zero. Frames that are valid are shown with a v on the *NETMON display. Bit 3 is set if a frame is incomplete (b for 'abort' on *NETMON), bit 4 if the CRC is incorrect (e for 'error' on *NETMON), bit 5 if the clock fails (d for 'data carrier detect failure' on *NETMON) and bit 6 if there is a receiver over-run (o on *NETMON). Bit 2 (idle) is set when the line goes inactive. This should normally happen at the end of a packet, so *NETMON displays an i and a newline ready for the next scout frame.

c It converts the serial bit stream from the network into 8-bit bytes, which the 6502 may then read from it along the 8 wires of the data bus. It also, of course, performs the necessary parallel/serial conversion for data going out.

d It assembles frames for the network and dismantles them when they arrive. Assembling involves transmitting the flag and station numbers before the data, computing the CRC, and adding the CRC and the closing flag at the end. Dismantling involves not only removing this packaging but also detecting errors, and informing the 6502 of them.

7.3 Disabling the Econet

The Econet software claims NMI in a BBC machine whenever it is not required by any other application. The only other application currently common in BBC machines that might lay claim to NMI is the disk filing system, and that only claims it while it needs to do so, which is during a disk transfer. Once the transfer is complete the claim is relinquished in favour of the NFS. When the NFS is given NMI by the OS it places code to deal with NMIs at &D00 and programs the 6854 to interrupt as soon as a frame is detected. The code at &D00 will check the destination station number and reprogram the 6854 either to read the transmission or to ignore everything until the next interrupt. This code stays there even if the NFS is disabled in favour of the cassette or disk filing systems, so the machine remains available to remote Econet software. Problems only arise if the user corrupts the NMI code at &D00, for example by running software that relocates there. Next time any frame passes along the net the machine will jump to &D00 and crash. The best method for avoiding

this problem depends on the application. We have evolved the following five techniques

a Disable NMI. This can be done by reading location &FE18 (in Basic, PRINT ?&FE18 will do). Note that the Econet software will re-enable NMI's if it is called, but the disablement is effective as long as the NFS is not asked to do anything.
b Instruct the 6854 not to interrupt. This can be done by setting the interrupt enable bits in its control register 1 to zero, or easily in Basic by ?&FEA0=0. As with **a**, asking the NFS to do something will reprogram the 6854 to interrupt.
c Poke a return code into location &D00, if this location is not wanted for another purpose. The correct code is RTI, &40, so ?&D00=64 will do.
d Unplug the Econet lead from the rear of the machine.
e Claim NMI with *FX 143,12,255.

The NMI jump is however not the only problem affecting those who might wish to run non-network software on a network machine. The BBC OS has special facilities to allow the REMOTE command on the Econet to intercept anything or everything the user might wish to do, facilities which may be hijacked by the more cunning software writers to protect or provide special effects in their products. One consequence of these arrangements is interception of keyboard operations in all Econet machines, whether or not the network is active. As Robin Newman first pointed out in the Acorn User (March 1984), the NET vector at &224 is accessed every time the RETURN key is pressed. This is harmless so long as it points to &FFA6, where there is an RTS. However, the NFS changes it to point to &FF36, and this causes problems with certain software. The solution is to poke location &224 with &A6 (&225 remains at &FF).

7.4 Frame structure

All information on the Econet is sent in packets or frames. I prefer the word 'packet' for all frames, but packet is often used to refer to a group of frames and we will have to stick with this convention. So a 'frame' is the smallest message unit (or packet!) that can pass along the network, and a 'packet' is a group of (between one and four) frames, which constitute a transaction. Loading a file from a fileserver will normally require a large number of such transactions, since the file first has to be asked for

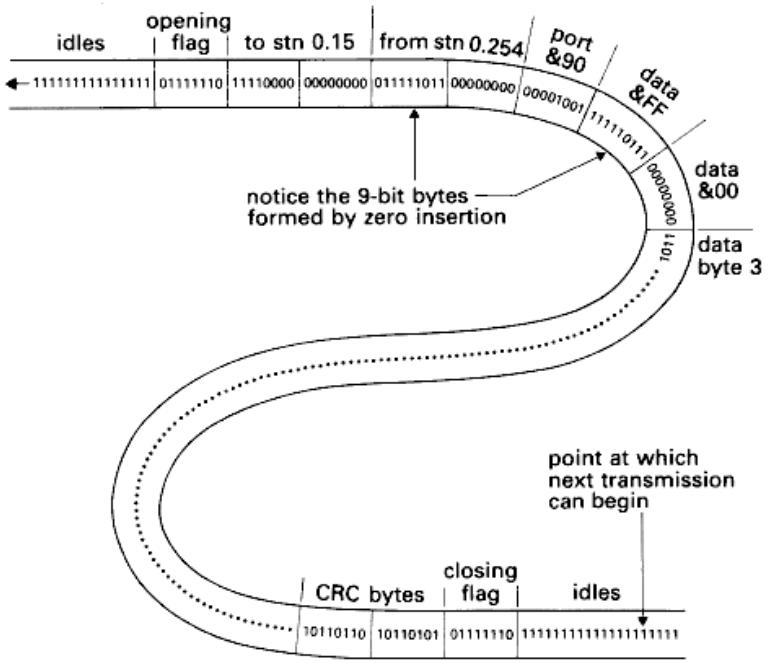

Figure 7.1 Frame structure

and then has to be loaded in chunks. Each frame has the following structure:

a An 8-bit opening flag, always 01111110.
b The destination station number, two bytes. The first byte is the machine number as set on its address links (?&FE18), the second is its network number - zero unless it needs to be addressed through a gateway or bridge.
c The transmitting station number, also two bytes.
d A 'port number', one byte, which is used to tell the software in the receiving machine the purpose of the message.
e A data field of any length. With the Econet clock at 240 kHz a 20 K high-resolution screen block should monopolise the network for two-thirds of a second - more in practice owing to overheads. The average block size on the normal network (excluding the large number of very short frames) is a few hundred bytes: there are very few of more than a few K. The *NETMON program cuts off the data field so that a

normal 4-frame fileserver transaction sits cleanly on one line. Note, incidentally, that a clock of 240 kHz is described colloquially as '240 K', and this means 240 000 bits per second or 240 kbaud, but the 20 K high-resolution screen is 20 480 bytes or 163 840 bits.

f A two-byte Cyclic Redundancy Check character, called the CRC for short. All the time the ADLC is sending out characters (including station number and control bytes, such as the port number) it is calculating a weighted sumcheck known as the CRC. This is an operation which happens to be much simpler for a small block of circuitry than for a human. It is merely a matter of dividing the whole frame (viewed as a long binary number) by a constant, and sending the remainder or its complement at the end. On reception the same calculation is done (by hardware, 'on the fly') and the CRC computed on reception compared with that sent at the end of the block by the transmitter. If they do not match, the data must have been corrupted at some stage and the CRC error flag is set.

g A closing flag, also 01111110. Until this is received the ADLC does not know the length of the frame. Since there are three receive registers the two CRC bytes will still be held in the ADLC when the closing flag is detected. They can therefore be checked and the Frame Valid or CRC error bits set.

7.5 Zero bit insertion

With variable-length blocks there must be a unique bit pattern to mark the end of a block, one that cannot occur in the data. The pattern 01111110 is the byte &7E, and a system that forbids this byte would not be very useful. CP/M has had problems because it required an end-of-file marker (the byte &1A), so this had to be a forbidden byte in data. ADLC's get over this problem by always inserting a zero after every occurrence of five consecutive 1's. This insertion is done just as the data is being put onto the network. Any zeros that follow five consecutive 1's are removed on reception. So six consecutive 1's followed by a zero must be a flag (flags, of course, are added to the front and rear of a frame *after* zero bit insertion). Seven consecutive 1's indicate an aborted frame, fifteen or more a free bus. One of the functions of a terminator is to pull the undriven bus into the continuous '1' state, so that any station on it can transmit as soon as it can count '1' for fifteen clock cycles. The '1' provided by the terminator is a mere 0.6 V, and is not quite the same as the 5 V '1' provided by a station that is transmitting 1, but it is adequate

for this purpose. Five volt? Well, that's what I thought for several years, since the Econet line is driven by digital circuitry that operates on signals that are at either 5 V or zero. However, the 75159 actually drives the line up to a volt 'inside the supply rail', so the levels may be 1 V and 4 V above Earth. For the other state, they will be 4 V and 1 V. The voltage difference on the lines may thus be 3 V, but the swing between '0' and '1' would be 6 V. These figures are approximate anyway, as there will be voltage drop on the network between the 75159 that is driving the line at any one instant and the measuring point. Meanwhile, 5 V will do.

8 The Econet circuit

8.1 Connection to the network

The Econet circuitry on the BBC machine occupies the top left-hand corner of the board - the diagram in Figure 8.1 follows the board layout. Looking into the socket from the outside, the left-hand pair of pins (numbered 1 and 4 on the DIN standard: the peculiar numbering is necessary to be compatible with 3-pin audio connectors) are the data lines while the right-hand pair (3 and 5) are the clock: the centre pin is Earth. The clock lines are input only, and are fed to half of IC94. The data lines are bidirectional. Output to them requires a line-driver chip, and this is the 75159 (IC93). It contains two line drivers although only one is used (the other would be used if the machine had a clock generator, as did BBC boards 1 to 3). The 75159 is enabled (activated) by the RTS (Request-to-Send) output of the 68B54.

A_0-A_{15}	Address bus lines 0 to 15
ADC	Analogue to Digital converter
CE	Chip Enable
CS	Chip Select (functionally equivalent to chip enable)
CTS	Clear to Send (input to tell 6854 it may send)
D_0-D_7	Data bus lines zero to 7
DCD	Data Carrier Detect
EOC	End of Conversion input on ADC
INTOFF	Interrrupts Off
INTON	Interrupts On
IRQ	Interrupt Request
LS	Low-power Schottky (embedded in IC reference number)
NMI	Non-Maskable Interrupt input on central processor
RTS	Request (or Ready) to Send (output from 6854)
RxC	Receive Clock
RxD	Receive Data
STATID	Station Identification
TxC	Transmit Clock
TxD	Transmit Data
ϕ	Clock

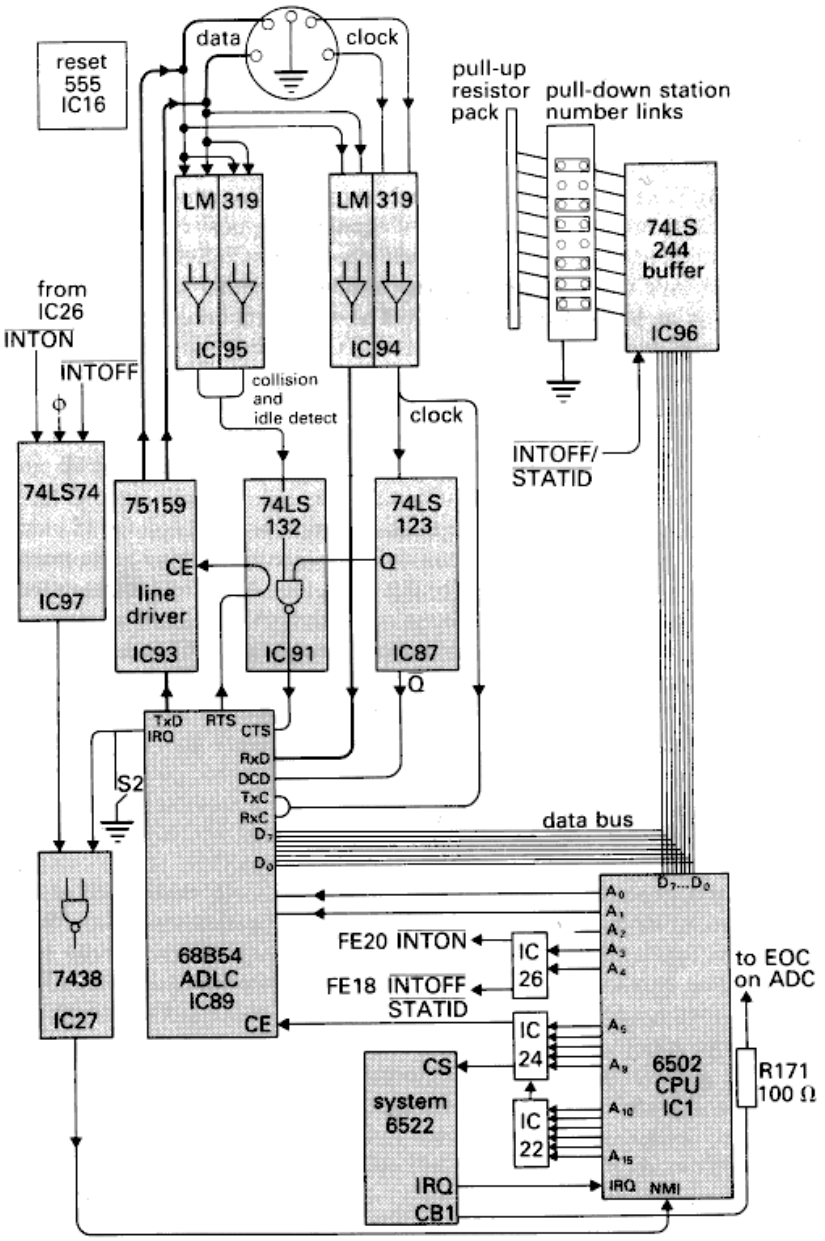

Figure 8.1 Circuit diagram

8.2 Input/output

Input requires something to detect a voltage difference (a comparator), because of the balanced line system used by the Econet (described in Chapter 6). The LM319 contains two comparators, which in IC94 are arranged to detect signals consisting of equal and opposite voltages on the inputs. One detects the clock and feeds it to the RxC (Receive clock) and TxC (Transmit Clock) inputs on the 68B54, the other detects the data and feeds it to RxD (Receive Data). The two comparators in IC95 are arranged to give a logic 0 (on the input to IC91) when the data lines carry valid data, which means opposite logic levels, and a logic 1 when they carry the same or an intermediate voltage. This output therefore means *either* there is a collision *or* the line is idle. If continuous 1s (either idle 1s or transmitted 1s) have been detected for more than 15 clock cycles *and* DCD indicates there is a valid clock, then transmission may start. Clearly transmission may start if there has been a continuous collision for 15 clock cycles, but (1) this should not occur anyway since all stations cease transmitting if there is a collision and (b) if it does, it will cease soon afterwards as the transmitting station will detect a collision. All the time a station is transmitting it monitors its 'Clear to Send' input from IC95, and as soon as this indicates a collision the transmission is halted. Transmission can therefore only take place when there is a valid clock *and* while there is no collision, which seems reasonable.

8.3 Addressing the ADLC

The central processor (6502) addresses the ADLC (68B54) at locations &FEA0 to &FEA3, though a study of the circuit diagram shows that there are seven further sets of four locations which will work equally well because that part of the I/O area called 'Sheila' is not fully decoded. Two address lines are fed straight to the ADLC, A0 and A1: this means the bottom two bits of the address bus are used to address different parts of it (i.e. it is addressed at &XXX0 to &XXX3). The next three address lines (A2, A3 and A4) are not connected to anything that concerns the ADLC so their values are irrelevant to it. Address &FEA0 thus has the same effect as &FEA4, &FEA8, &FEAC, &FEB0, &FEB4, &FEB8 and

&FEBC. The chip enable of the ADLC comes from pin 10 of IC24: this is only activated if

a address lines 5–9 carry the pattern 10101; and
b address lines 10 to 15 carry the pattern 111111.

This is because lines 10–15 go into IC22, which tells IC24 when they are all 1s. Lines 5–9 go into IC24 which, given the pattern 10101 on these pins *and* a 1 coming in from IC22, activates output pin 10 which is connected to the ADLC. IC22 thus restricts the address range to &FC00 to &FFFF (&FC00 is 1111 1100 0000 0000) while IC24 brings it down to &FEA0 to &FEBF.

8.4 Interrupting the 6502

The Interrupt Request output of the 68B54 is triggered every time a frame is detected. It is connected (when allowed by S2 and IC97) to the Non-Maskable-Interrupt on the 6502. Net interrupts must be dealt with very fast, becase the ADLC has only three receive buffer registers and with a network clock rate of 300 K (300 kHz, 300 kbits/s or 300 kbaud) three bytes arrive in 80 microseconds. This is the time the 6502 has to decide whether to read or to ignore the message, and if the net had to compete with all the other interrupts going on in the BBC machine there would be many missed packets. When the 6502 receives a NMI it jumps to the address given at location &FFFC. In the BBC machine &FFFC is in the MOS ROM and contains the address &D00, so the 6502 jumps to &D00. In the BBC machine only one filing system at a time is allowed to claim NMI and have code at &D00: when the net is active this is the NFS so the code at &D00 is dedicated to dealing with an interrupt from the 6854. One other filing system currently uses the NMI – the DFS. This is because the DFS also has to react fast to an external event – the rotation of a disk. If an Econet machine is also a disk machine, then while the disk is actually active the DFS claims NMI from the NFS and puts at &D00 the code needed to deal with a disk interrupt. As soon as the disk transfer has completed the Econet is allowed to reclaim NMI and replace the disk NMI routine with its own. While disk transfers are taking place a BBC machine will ignore all Econet activity.

8.5 The station ID

Eight links beside IC96 are used to set the station ID, as a binary number in the range 0–255. These merely pull down to Earth a set of eight wires, which are pulled up to +5 V by the eight resistors in Resistor Pack 2: the resistor will pull up any line where a link is not present. The normal chip in IC96 is a 74LS244 which makes the bits 1 if a link is not present (so the station number is 255 if all links are out). If the 244 is replaced by a 74LS240 then the bits are 1 if a link *is* present (so the station number is zero if all links are out).

The links are read at address &FE18: when they are read, interrupts are turned off. Interrupts are turned on again when &FE20 is read. This allows software to turn off NMTs - a rare feature!

8.6 The Remote Subroutine Jump

One facility the Econet provides is the Remote Subroutine Jump - one machine can cause another to jump to a specified location to execute whatever subroutine is there. This needs a little hardware assistance, because though the Econet ROM is active when the RSJ is requested the jump must take place after the Econet software has finished and returned control to the MOS or to whatever was happening before the Econet interrupt came in. The Econet software has to lay a 'time-bomb', which must cause the RSJ as soon as it has relinquished control. If the Econet software initiated the RSJ directly then the routine requested by the remote station would be running within the original Econet interrupt (i.e. within the interrupt that occurred when the packet requesting the SJ arrived). This would severely limit what the subroutine could do - hence the need for the time-bomb.

The bomb is activated by programming IC3 (the system 6522, a VIA or Versatile Interface Adaptor) to load a byte into its shift register from pin CB2. Loading a byte into this register takes time as it is shifted in bit by bit: the point is that this shifting in is done by the 6522 all by itself, so while it is being done the central processor can get on with finishing off the Econet interrupt. Pin CB2 happens to be connected to the video ULA, but it does not matter as the contents of the byte shifted in are immaterial. Unfortunately the 6522 was designed to send out timing pulses on CB1 whenever it is asked to shift bits into CB2 and it cannot be prevented from so doing. CB1 happens to be connected to the End-of-Conversion output of the A/D converter, and sending bits into this output

could cause trouble. Hence resistor R171, 100 Ω, put in this wire so that the wrong-way signal does no harm.

Setting the VIA to shift the byte in from CB2 is only half the job. The Econet software also sets it so that when the shift has finished it will interrupt the 6502, and arranges the IRQ mask and other code so that this interrupt causes the RSJ to take place. While the data is shifting in, the Econet goes to bed: as soon as it has done so the shift-in finishes and the software to which the Econet has returned control gets interrupted so the RSJ takes place. Ingenious!

9 Problems

One of the main arguments for teaching computing in schools is that it provides excellent training in problem solving: computing without problems is like politics without arguments. The Econet of course throws up its fair share. The worst are those that jam the net, leaving all users feeling rather as motorists do when their engines refuse to start. Locating and removing the cause of the jam is not always easy because the network is a bus common to all the machines and the fault could often be anywhere.

9.1 Laying the network

a Great care needs to be taken in the initial laying: careful stripping of the cable so there are no nicks in it, high quality soldering and the avoidance of any potential for the development of open or short circuits are all sound investments for the future.

b Junctions, as described in section 6.5, should be put in at intervals to allow sections of the network to be rapidly isolated from each other.

These precautions should reduce the number of intermittent and very-difficult-to-trace problems.

9.2 Additional protective measures

Five additional protective/detective measures could be taken:

a Surge suppression diodes can be put in at intervals, to remove spikes. Suitable ones are RS 283-255, wired with the negative end on the earth and the positive end on the data line. With the old Acorn system 283-255 will do for both data and clock lines: with the newer systems the extra power on the clock lines requires the higher voltage diodes (RS 283-261).

b A spike detector could be added, to give a clue as to what might have caused a fault.

c Two-way opto-isolators could be installed. These are currently merely circuits drawn on the back of an envelopes, which would have four separate one-way amplifier/opto-isolator circuits to deal with each signal in each direction. They would have to sample the line at a frequency of at least ten times the clock rate.

d A split bridge can be used to isolate sections of the net from each other. A bridge conveys frames from one network to another, and there is no need for the two to be adjacent. The two halves of a bridge can be linked in various ways, and can be millimetres or kilometres apart. This offers opportunities for opto-isolation, and also good Econet prospects for split sites.

e All machines on the network could operate from an independent mains supply, equipped with proper spike suppression. It should be noted, incidentally, that ordinary spike-suppresser plugs are hardly necessary with BBC machines as the switch-mode power supply already incorporates a fair degree of immunity to mains-borne interference.

9.3 The 75159 line driver chips

When lightning strikes near an active Econet there are induced current surges both in the Econet cable and in the mains. This normally blows up some of the 75159 line driver chips. Machines with bad line drivers generally cause a 'line jammed' message on all other machines on the net. It is not yet certain whether the problem with the 75159's is due to pickup from the mains or from the Econet cable, but the problem only arises with machines that are both switched on and connected to the net. 75159's are easily replaced on most machines because they are in sockets, but there is a batch of Issue 7 boards where the 75159's are soldered in. 75159 chips sometimes go just a little bad - working normally but making the whole net just a bit off colour. The symptoms in such cases are similar to those caused by T-junctions over a couple of metres, bad termination or a clock that is too slow or too fast, and are

a slow net operation. If a particular operation between X and the file-server takes 5 seconds when station Y is plugged in and 1 second when Y is unplugged, then, assuming nothing else is using the net, Y may have a bad 75159;

b occasional idles, which can be detected by the Acorn *NETMON program. An idle is when a flag is detected on the net but no packet follows it;

c bad voltage levels, and poor signals, on the clock and/or data lines. The correct voltage levels on and between the lines depend on whether

you are using the Acorn or the SJ clock/terminator systems, and an oscilloscope is needed to measure them;

d a machine is found to cause net problems only when it is turned off. This fault can remain unsuspected for a very long time if the machine is one that is normally left on.

Sometimes these faults can be traced to a bad LM319 and sometimes to something deeper in the machine, but the prime suspect in most cases is the 75159.

9.4 Miscellaneous hardware problems

The easy hardware problems that may arise are

a 'No Clock' - which generally means your machine has become unplugged from the net. It may of course mean that the clock lines are shorted, or even that there is a problem with the clock box - someone may have unplugged it to plug in their cassette recorder.

b 'Channel' generally arises from the same reason as 'Not listening': you have run an Acornsoft game or a relocated program in your machine, but it can qualify as a hardware-generated fault because it can also arise if you have turned your machine off and on again. The solution is a fresh log in - see 'Not listening', p. 80.

and the more difficult ones are

c 'No reply', which means your transaction was accepted by the fileserver (or other machine) but has not been completed. Something serious has happened in between, but it is worth another try in case it is simply a heavily loaded network.

d 'Net error' often means incorrect wiring: if the wiring is correct it may mean the clock speed is unsuitable or the termination is bad - perhaps a terminator has been unplugged or you have just extended the network. If the wiring, clock and termination are satisfactory you have a similar problem to 'Line jammed'.

e 'Line jammed' means there appears to be continuous information on the lines, but this is most unlikely to be genuine data. There are many possible causes: the five most common are

(i) A machine with a dud 75159 is on the net - or if there has been a thunderstorm recently, perhaps a dozen machines with dud 75159's. Other chips that can be affected are the 68B54, a LM319 (particularly the right-hand one, which is connected to both clock and data lines) or even the system 6522, but these rarely crash the net - they simply make their own machine inoperative. As mentioned on p. 77, if these chips are only partly faulty they may merely slow down network operations.

(ii) Someone has plugged an Econet lead into the cassette socket of a machine.

(iii) A short circuit has developed on the line, probably in one of the Econet leads to a machine.

(iv) As for 'Net Error', a problem has arisen with the clock, the wiring or the termination.

(v) A machine has developed a serious fault in its network software or hardware. This is only likely if you have experimental, home-made or elderly machines on your network.

Locating the fault requires a lot of unplugging and plugging in: it is simply a matter of unplugging everything (except clock, a fileserver and terminators) until the fault clears and then plugging everything back in again until it recurs. A fault in a clock, fileserver or terminator is more difficult to locate without spares, or substantial expertise with an oscilloscope.

9.5 The earthing problem

Finally, the worst hardware problem. The BBC machine has a switched-mode power supply, and there are as part of it capacitors between earth and power. If the earth is not connected these capacitors will pull the machine's earth up to a hundred volts or so, which will not harm it so long as it can float there and remain unconnected. The machine will operate quite happily with the earth at 100 V so long as the +5 V line remains at +105 V, and the user need never know since the keyboard is plastic. It is of course necessary for the monitor to be disconnected from earth too, but this can be arranged by plugging both computer and monitor into the same 4-way distribution block and then breaking the earth into that block (or by using a monitor that has no earth connection). If it is an ill-used block of poor design and indifferent wiring, plugging something else (e.g. a cassette recorder) into the same block at the end where the wire enters could break the earth momentarily (or for longer). As mentioned earlier, there need be no problem if the machine is not connected to a network - though the user may get a shock if he touches metal parts on the monitor. The fact that the machine's internal data lines are all at 105 V above earth need cause no trouble - unless the machine on

which this unlikely chain of events occurs is connected by a four-wire lead to a network.

Needless to say, this combination of circumstances has occurred. All the other machines on the network received 105 V where they expected five, which was expensive.

9.6 Software-generated problems

A couple of problems in the list on pp. 77–78 can be caused by software rather than hardware, but the following are strictly software. There are nearly 100 error messages listed in the Econet handbook and the problems causing most of them cannot be cured by the user. Only the most common problems are mentioned below.

a 'Not listening' generally means you are trying to access a fileserver that does not exist, probably because you have run some software in your machine (anything that relocates down to &E00) that has corrupted the location where your default fileserver number is stored. A solution is to specify a fileserver number in the *I AM command. The next most likely possibility is that you have given the wrong fileserver number in the *I AM command – perhaps 245 instead of 254. The third possibility is that the fileserver is actually not listening – perhaps someone is using it for something else. A fourth possibility is that heavy traffic on the net has prevented your message from getting through, so you could try again.

If your problem arises from relocated software, the safest way to get going again is to do a CTRL/BREAK and then log in afresh. If you do not wish to do this you can specify a fileserver number in the *I AM command, but if you do not specify the high (or network) byte as well you may still get 'Not Listening'. Your machine stores your currently selected fileserver number in locations &E00 and &E01 (in the I/O processor, of course), &E00 holding the number you normally use and &E01 the network number. This should be zero unless you are using a fileserver which is connected to your network through a bridge or a gateway. In the *I AM command you can specify the network number thus (as described in section 4.2a)

```
*I AM 3.254 USER
```

to access fileserver number 254 on network number 3. Your machine then stores 254 in &E00 and 3 in &E01 and subsequently uses these

values whenever it performs any fileserver transaction. If you run software that, unknown to you, alters location &E01 to (for example) 67 then the command

```
*I AM 254 USER
```

will attempt to access the default server on network number 67 and you are therefore likely to get the reply 'Not Listening'. It is of course not good BBC manners to address these locations directly, but if you do get 'Not Listening' a quick check on their contents may explain why.

b 'User not known' can arise when
(i) you attempt to log into the wrong fileserver, or
(ii) Shift-Break is pressed and there is no user called BOOT on the currently selected fileserver, because Shift-Break causes a machine to send out a *I AM BOOT message. A properly managed system will of course have a user called BOOT and a BOOT directory set to provide a helpful starter message.

c 'Not found' can arise if a directory has a boot option set (with *OPT 4) and there is no file !BOOT in it. The error can be puzzling because it occurs immediately after you have logged in with *I AM FRED or something, giving the false impression that the login has failed. The message can also arise on the SJ fileserver if you refer to someone else's files through a Private directory.

d 'Insufficient access' can arise on the Acorn fileserver if you try to refer to your own files through the root rather than directly. If you log in with *I AM FRED and have a file X with access string WR/R, you can load it with LOAD "X" or with LOAD "$.FRED.X". If the access string is WR/ you cannot use the second version. No-one in his right mind would be using such unnecessarily long commands anyway, but they can be generated by software and the error may be puzzling when it occurs.

Appendix 1 The Z80 on the Econet

To use the Econet, a computer needs the necessary hardware and software. The hardware is relatively easy: it takes hours to design, days to build and weeks to test. The software is not so easy: it takes weeks to design, months to produce and years to test. High-level software such as fileservers takes months to design and years to produce and test. Econet hardware has been implemented for the Apple II, the Nascoms 1 and 2, the Research Machines 380Z and 480Z, S-100 bus machines and all Acorn computers, but only the last four are commercially available. Torch computers also offer a version of Econet called Torchnet, using BBC hardware with the Torch Z80 and Torch software. Torchnet is a completely different approach: any Torch machine can access the disks on any other. A similarly independent Econet (though 6502 based) is the E-net marketed by GSL. It also has special software, which is deliberately incompatible with the Acorn net - optimised for classroom use of a single Winchester fileserver. The SJ software, though running on totally different hardware (Z80 and SIO), is written so that SJ and Acorn products can freely intercommunicate on the net: it can run with CP/M 2.2 or independently of it.

The SIO

Interfacing a Z80 to the 6854 used as the network controller in the BBC machine has been done on a Nascom-1, but it was not easy because the 6854 was designed to connect to 6502-type central processor chips and Z80 boards do not normally have the required two-phase clock. It is rather more satisfactory to use the SDLC chip made for Z80 systems, called the Z80 SIO (for Serial Input/Output). This is better than the 6854 in that it can do its own checking against the station number so need only interrupt its CPU for messages actually directed to it, but it lacks the ability of the 6854 to count the 15 1's needed to establish that the line is clear and it cannot 'turn the line round' as fast as the 6854 without extra hardware.

The hardware

The Torch and GSL Econet interfaces are standard BBC ones - only the software is different. The 380Z/480Z/S-100 interfaces have the same line driver, receiver, collision and idle detect circuitry as the BBC - which at least ensures electrical compatibility - but the ADLC functions are performed by the SIO chip. Only about half the hardware on the 380Z board is required for the Econet: the rest is a 64K RAM expansion and a ROM socket, with the necessary decoding for the RAM to be paged as required. One important use for this paged RAM is when the 380Z is acting as a fileserver - the RAM on the Econet board then provides cache memory which considerably speeds up fileserver operation.

The software

When running under CP/M the software appears simply as an extension of the BDOS calls. In CP/M 2.2 these run from 1 to 36: the Econet primitives add calls from 64 upwards. The Econet primitives are supplied as command file NET.COM or ECOZ.COM which on loading

a relocates itself upwards to just below the CCP;
b checks the existence of the hardware by loading the SIO registers and reading the station ID;
c sets up receive control block facilities;
d sets up a vector to intercept BDOS calls; and
e returns to the CCP.

Once active, the Z80 system can act as a normal client on the network. *commands entered to the CCP (or supplied by applications software) will be processed by the net software and all the fileserver *commands given in Chapter 4 can be used as if from a BBC machine. The currently selected directory is addressed as drive P, and CP/M commands such as DIR P: provide a directory listed in the CP/M way while Acorn commands such as *CAT or *. provide a listing laid out the Acorn way. Z80 systems with disks can load a Level 1 fileserver which makes their disks accessible from elsewhere. Much better is the Level 2 fileserver, though this cannot be used to access CP/M disks as it requires the special disk layout necessary to provide the full Unix-type passworded hierarchical directory structure described in Chapter 4. The Level 2 fileserver includes

the NET.COM primitives and is the 'proper' fileserver: the Level 1 fileserver is not recommended for serious use. SJ also produces special hardware for the fileserver: a complete 'black box' including Winchester, random-access tape backup and 256K of cache memory and a miniserver based on similar hardware but with a floppy disk interface.

Appendix 2 Glossary of terms

6502 The 6502 is the central processor chip in the BBC machine, the Atom and Systems 3/4/5, one of the second processors, the Apple, the Pet, the Vic 20, CBM 64 and many others. The processor in the BBC machine is the 6502A - the A meaning it operates at a clock speed of 2 MHz. The BBC second processor uses a 6502C (3 to 4 MHz).

6854 This is a 1MHz ADLC.

68A54 This is a 1.5 MHz ADLC.

68B54 The 68B54 is a 2MHz ADLC. This means it will operate in conjunction with a CPU chip that is operating at 2 MHz. In fact 6854's and 68A54's have been known to operate in BBC machines, but not reliably. The 6854 is sometimes referred to as the MC6854, because it is made by Motorola.

ADLC The Advanced Data Link Controller (ADLC) is the silicon chip on every Econet machine that organises the frames or packets on the network. Since it can be used to implement a synchronous (='isochronous') data link following the IBM standard, it is a SDLC chip. See Chapter 7.

Anisochronous/Asynchronous One convention applies the terms synchronous and asynchronous to the terminal devices (computers) and isochronous and anisochronous to the transmission system (the cable and associated control systems). In asynchronous transmission no clock is provided: each byte needs start and stop bits and the two stations must have pre-arranged the clock rate. In synchronous transmission a clock may be pre-arranged, though it should be provided with or deduced from the data. Bytes do not require start and stop bits, though blocks of data may need to be preceded by synchronising characters. If the transmitting station has no data to send it must send synchronous idles to keep the receiving station in step, or terminate the block.

Autobooting This occurs when a BBC machine loads and runs software when it is turned on (or when BREAK is pressed). On the BBC

machine this is arranged *either* by holding down SHIFT during the power-on or BREAK operation, *or* by connecting link number 5 on the keyboard. An Econet machine sends out a *I AM BOOT command to the fileserver, and if there is a directory called BOOT there with a file in it called !BOOT *and* the boot option on the directory is set correctly with *OPT 4,n, then the software designated in the !BOOT file will be initiated.

Balanced line A balanced line cable consists of a pair of wires on which the signal is such that the voltage on one wire is always equal and opposite to that on the other: when one swings positive the other swings equally negative. The signal is detected by comparing the voltage levels on the two wires. The zero point about which the voltage swings need not be at Earth potential.

Baseband Baseband describes a digital network which carries only binary bits at one specific frequency: the alternative (which generally has more capacity but at greater cost) is Broadband. Multiplexing on a baseband network has to be time division.

Baud For Econet purposes, 240 kbaud means the clock operates at 240 kHz or 240 thousand cycles per second: one bit is sent per cycle so at this rate 30 kbytes are sent per second. Baud actually means data transitions per second, and does not equal bits per second if a data transition carries more than one bit of information (it is unlikely to carry less). The Econet can operate with BBC machines at rates between 100 and 300 kbaud. There is an important distinction between k (=1000) and K (=1024) here, but this is often ignored: the clock rate mentioned above is normally described as '240K'.

Bit-serial Bit-serial means the eight bits of each byte are sent one after each other along a single cable. Bit-parallel means the bits are sent in parallel along 8 parallel wires. Communications lines must operate in serial mode, because copper is expensive. Inside a computer parallel mode is quicker.

Boot option A boot option is set for a disk or for a network user with *OPT 4,n, where n can be 0, 1, 2 or 3. Zero turns off autobooting. 1, 2 or 3 make the system *LOAD, *RUN or *EXEC the file !BOOT whenever the disk is entered with SHIFT/BREAK or whenever the network

user logs in. Note that SHIFT/BREAK on a network causes a login as user BOOT.

Bridge A bridge is a junction between two networks of the same type. In all other respects it is the same as a gateway – see p. 89.

Broadband Broadband describes an analogue network, which may allow many frequencies on it – each for a particular communications channel. The actual signal on the cable is analogue, but it can carry many digital channels. Broadband communications involve frequency division multiplexing. Broadband implies radio-frequency communications with data modulated onto a r.f. carrier.

Buffering This means storing data temporarily between source and destination so that it can be sent when required, at the speed or in the packaging required.

Bus A bus is a common set of wires shared in parallel by a number of electrical units. Some buses have over a hundred wires, some only a few. The Econet is a four- or five-wire bus.

Case independence This means that the system does not distinguish between the files Fred, fred and FRED, although directories do record the difference. On the DFS and on earlier fileservers a file called FRED cannot be renamed as Fred by the command *RENAME FRED Fred. It is necessary to give two rename commands: *RENAME FRED X and *RENAME X Fred. The second processor, the Winchester and the SJ fileservers do allow the single command: they will allow FRED to be renamed as Fred with the command *RENAME Fred Fred. Case is irrelevant in all operating system commands (everything following a *), which may be in upper case, lower case or a combination. To get the same effect in your own programs it is necessary to eliminate the bit (value 32) that distinguishes between upper and lower case: this can be done by ANDing each character with &DF, which is 11011111.

Clock A clock is a small circuit attached at one point on a network which puts a 5 V square wave of frequency between 100 and 300 kHz onto one of the pairs of wires in the cable. All operations on the net are synchronised by this clock.

Collision arbitration algorithm This is the software that decides how long each machine should wait before retrying the line after two or more have tried to transmit simultaneously. If the delay is made to depend on the station number then no two machines can retry at the same time, which is what the algorithm should ensure.

CSMA/CD CSMA/CD stands for 'Carrier-Sense Multiple-Access with Collision Detect', which in the Econet's case has the following meaning. Carrier-sense means machines check for an idle line before attempting to transmit. Multiple-Access means all the machines have equal access and may attempt to transmit simultaneously. Collision Detect means that each machine listens to the line while transmitting and stops immediately if it senses a collision (which should not occur after the first few bytes because no other machine should start transmitting while another is doing so).

DIN sockets The German equivalent of the BSI is DIN. Their standards for audio connectors (for tape recorders etc.) are widely accepted and available. 180° refers to the fact that the 5 pins in the sockets used for the Econet occupy an exact semicircle.

Econet 'Eco' means 'to do with the home', after the Greek word for home, 'oikos'. 'Economical' is derived from oikos, though 'Eco' can be stretched to mean 'local'.

Empty slot Empty slot networks are generally rings around which packets are always circulating whether they actually carry anything or not. A station wishing to transmit cannot create a packet the way an Econet station does: it has to wait for an empty one to come by and then it may fill it. The empty packets are created by a special monitoring station. The advantage over the Econet's anarchy is that empty slots are guaranteed to occur quite frequently because (amongst various techniques) stations are not allowed to refill packets they empty.

E-net This is a special Econet marketed by GSL. It uses the same hardware as the Acorn net but its own software, including its own primitives and its own net protocol. E-net and Acorn Econet computers cannot communicate with each other. As a design feature communications are only allowed between clients and the fileserver: communication between stations or to other servers would require special software.

Ethernet Ethernet is the 'Big Daddy' of a whole family of CSMA/CD networks. It was developed in Hawaii for many island-based computers to communicate with each other on a single radio channel.

Frame This is the minipacket on the Econet. A packet sent from a station to the fileserver would normally require a scout frame (are you there?), an acknowledge, the data frame and a final acknowledge (section 4.5). Most immediate operations require only two frames, and the broadcast requires only one.

Gateway A gateway is a junction between two networks of different types: the description that follows applies equally well to bridges which are junctions between two networks of the same type. All packets contain 8 bits on every station number which are reserved for Gateway use. Station numbers really consist of two sections - a low byte followed by a high byte - the latter being always zero at present. It can be set in the *I AM command using a decimal point to separate the two sections, so *I AM 3.254 FRED will set the high byte to 3 and all messages sent with this setting will be ignored by all stations except gateways. If it passes by a gateway which has a connection to network number 3, the gateway will grab it and pass it through. When it eventually (or immediately) reaches network number 3, the gateway on 3 that inserts it will zero the high byte, so machine 254 on this network will accept it.

Hierarchical See Figure 4.1, p. 17.

ID ID means Identifier, used here as Station ID for the station number on the network.

I/O processor When a second processor (6502, Z80 or 16032) is added to a BBC machine the added machine is called the language processor while the original machine is the I/O processor: it handles all the Input/Output while the add-on does the main job. Since both share the same sort of memory addresses (at least from 0 to 65535) there has to be a way of directing information from outside (e.g. disk or net) into the most appropriate machine. The way chosen is to give the second processor (let's call it the 2P) priority from the bottom, and allow the I/O processor to occupy the very top 64K of the 4.3 gigabyte address range of the largest 2P available. In practice this means that addresses from &00000000 to &FFFEFFFF go into the 2P while those from &FFFF0000 to &FFFFFFFF go into the I/O P. &FFFEFFFF is over 4 thousand million bytes, which is enough RAM for any micro this year.

IRQ The Interrupt Request (IRQ), pin 4 on the 6502 chip, is normally held at 5 V by R85. When pulled down to Earth by one of the many devices that are connected to it, and if interrupt requests are currently allowed, the processor preserves the program counter and status register on the stack and jumps to the address held in the top two bytes of the OS ROM (locations &FFFE and &FFFF): this address is &DC1C and there are instructions to deal with the interrupt and jump onto &202 or &204 (BRKV or IRQ1V).

Isochronous This means synchronous – see anisochronous.

LAN LAN means Local Area Network.

NETV Net Vector (NETV) is an address held in locations &224 and &225, and it can be set to allow interception of all operating system calls. Its primary purpose is to assist the network *REMOTE command to take over control of the machine. Such potential power can be directed to many other uses, most of them for the benefit of the software writer.

N-MOS This means Negative channel Metal Oxide Silicon. Other chips may be P-MOS or CMOS (Complementary MOS – a mixture of both N and P, expensive but low on power consumption so used in battery-operated equipment). These are simply three of dozens of different ways of making silicon chips: N-MOS happens to be the way chosen for both the 6502 and the 6854.

NMI The Non-Maskable Interrupt (NMI), pin 6 on the 6502 chip, is normally held by R81 at 5 V. When pulled down to Earth by something else (in the current BBC machine only by the disk or Econet controller chips), the processor then preserves the program counter and status register on the stack and jumps to the address held in the ROM in locations &FFFA and &FFFB; this address is &0D00 and the NMI routine stored there then deals with the disk or Econet event that caused the interrupt. NMI's, unlike IRQ's, cannot be ignored.

OSCLI OSCLI refers to the Operating System Command Line Interpreter, and is a keyword in Basic 2 which allows a Basic string to be sent as a command to the OS. Page 463 in the User Guide describes how to use OSCLI from Basic 1.

Packet A packet is a frame or small group of frames comprising one transaction on the network – see section 7.4.

PAGE On BBC machines, PAGE is the marker for the bottom of free user memory. It is pushed up by systems such as disk or network. If PAGE is higher, space for your program is less. In lower case, a 'page' of memory is 256 bytes.

Primitives These are a set of machine code routines in each machine that organise the traffic between the computers so that programmers need not bother with details involving the ADLC's.

Protection mask This is six bits set or read by Osword &13 (as described in section 5.13) or by library routines *PROT and *UNPROT.

Protocol Protocol means an agreed system for sending/receiving information: a protocol defines the structure of the packets, when and how they are to be transmitted and what is to be done in the event of various sorts of failure.

Relocation Relocation of programs is an unfortunate habit adopted by software writers to provide themselves with more RAM. The disk filing system on a BBC machine takes up 2.75K of RAM, pushing PAGE up from &E00 to &1900, though files can be saved and loaded with PAGE at &1100. The Econet filing system takes up 1K, pushing PAGE up from &E00 to &1200, though files can be saved and loaded with PAGE at &1000. When Econet is added to disk there is some sharing of the workspace, so only another half K is taken up (PAGE goes to &1B00). This filing system workspace reduces the space available for user programs, so many of them, after loading, move themselves down over the filing system workspace before running. This is called relocation, and means the filing system needs to be restarted with a BREAK before it can do any further operations. Relocation causes all sorts of problems, especially as few software writers reset all the necessary pointers, and is to be strongly discouraged.

SDLC SDLC is the IBM Synchronous Data Link Control protocol for sending packets on a network.

SIO The Serial Input/Output chip (SIO) is the ADLC for Z80 systems.

Spooling Spooling means sending printer output to a temporary disk file, so that it may be printed out at some convenient time later. A spooling printserver can accept print output from dozens of stations simultaneously. It 'spools' all the 'print streams' to disk until the printer is free, and then unloads them, in turn, off the disk to the printer. It is clearly a useful thing to have on a network.

STDM Statistical Time Division Multiplexing (STDM) is TDM but with stations allowed to take time on the cable in proportion to their need. Simple TDM allocates fixed time slots to each station regardless of whether it needs them or not.

Synchronous See anisochronous.

System System computers are a now obsolete Acorn product. Acorn started with a System-1: the range reached System-5 when the BBC machine came out. The System-5 is roughly equivalent to a BBC machine with 6502 second processor, but is more expensive and not as useful.

Termination Termination is required at each end of the network, (a) to prevent the signal being reflected off the end of the cable and interfering with itself, and (b) to bias the data lines so that data plus is about 0.6 V above data minus when there is no computer driving the net - this allows the data lines to be read as continuous ones. For the first function a 100 Ω resistor is all that is required, so that the cable presents a constant impedence of 100 Ω to every station. For the second some power is needed. To avoid the problem of supplying power to both ends of a network as well as to the clock, newer systems have a clock which puts power on the net by raising the clock lines up to 5 V above the data: the terminators use this power to bias the lines as described above.

TDM Time Division Multiplexing (TDM) is one way of describing how a Baseband network such as the Econet shares out the cable between different stations - each takes it for a few milliseconds, rather as a holiday home may be timeshared between families. The alternative is FDM or Frequency Division Multiplexing, which is how a Broadband network operates - each user operating at a different frequency. The holiday home analogy is each family having a different room or floor, but all the time.

Token-passing This is another way of sharing out a network's time between stations to avoid the chance of any machine hogging it for too

long. Each machine is given control of the network in turn, usually whether it wants it or not, by a master machine.

Torchnet Torchnet is the network system supplied for Torch computers: it is Econet compatible with Torch extras.

Unix Unix is a minicomputer operating system developed at Bell labs, and now available on a number of 16-bit microcomputers as well. It is based on the language C but can run a large number of languages: its hierarchical directory structure inspired the Acorn and SJ fileserver structures.

Z80 Z80 is the central processor chip in the 380Z, Nascom, ZX80, ZX81, Spectrum and many other computers.

Appendix 3
Further information

1 Addresses

Acorn Customer Services and Technical Enquiries, Units 8-9, Technopatk, Newmarket Road, Cambridge CB5 8RB.

Anixiter (UK) Ltd, 15 Chesford Grange, Woolston, Warrington, Cheshire (tel: 0925 810121), supply cable suitable for the Econet.

Author C.C.H. Dawkins, Felsted School, Dunmow, Essex CM6 3JG.

BICC (British Industrial Cables and Connectors), Helsby Works, Helsby, Warrington WA6 0DJ (tel: Helsby 2700), supply cable suitable for the Econet.

GSL (Geophysical Systems Ltd), West Portway Industrial Estate, Andover, Hampshire SP10 3SG.

RS Components: they sell only to institutions, not to private buyers, but their products are available from most dealers.

SJ Research (Econet specialists), 108 Mill Road, Cambridge (tel: 0223 69927).

2 Bibliography

This Micro Guide has been completed with reference to only one Econet publication: the Acorn Econet System User Guide produced by Acorn in November 1982. Seven other Econet publications came out in 1984:

Econet level 1 Fileserver User Guide
Econet level 2 Fileserver User Guide
Econet level 1 Fileserver Manager's Guide
Econet level 2 Fileserver Manager's Guide

Econet Advanced User Guide
USPEC 32e
Networking with the BBC Microcomputer

The first five are Acorn manuals, between them replacing the original User Guide. They provide a much more flexible resource than the original combined guide, but the Econet Advanced User Guide does not go into as much detail on communications between clients and a fileserver.

The Council for Educational Technology, 3 Devonshire Street, London W1N 2BA, publish a free 'User Specification', number 32e, on Networks.

'Networking with the BBC Microcomputer' by R.G. Napier, Prentice-Hall, 1984, provides an excellent introduction to the Econet, expanding on and explaining most of the contents of the first four Acorn Guides.

Index

DNFS,
NFS 360 ?88304 = default printer server station number.

Econet is the Network system produced by Acorn Computers Ltd for use with the BBC and other microcomputers. This expert guide covers all that is pertinent to a real life installation of the system. The explanations are given in depth but with painstaking regard to clarity. Supported by examples and illustrations, there is detailed coverage of the general principles, hardware, differences between versions, the Econet software, circuits, and the problems that can arise and how to deal with them.

The author is a teacher who has been operating an Econet testbed for several years, with one kilometre of cable connected to equipment of all manner and description liberally distributed around his school. His experience with the realities of Econet is second to none and has enabled him to pass on many tips. His pleasant and lucid style and his unassuming but sure mastery of his subject make this Micro Guide both an invaluable handbook and a delight.

ISBN 0 582 36215 6

Longman