8 PAGED ROMs

The Acorn Electron and the BBC microcomputer both support the concept of a number of ROM based programs being resident in a machine in the same address space. Each ROM is *paged* in as required and then *paged* out as software in another ROM is required.

Paged ROMs work broadly in one of two ways. They act either as languages such as BASIC and LISP or they act as utilities such as filing systems and device drivers. Languages may also include such things as word processors and CAD graphics packages.

At any one time only one language should be active. Thus most Electrons will enter BASIC as the default language. The current language has access or control over the user RAM which it in turn may allocate to users e.g. for BASIC programs or word processing text.

While the one language is active any other ROM offering a service may be called upon as is appropriate, When a request for a service is generated the operating system interrogates each paged ROM in turn until the request is acknowledged and acted upon. Different types of request are indicated to each ROM by the operating system entering the service entry point of that ROM with an accumulator value representing the reason. These calls are called *paged ROM service calls*. If the service entry point is entered with A=7 this indicates that someone has asked the operating system for an OSBYTE call which the operating system failed to recognise and so is now asking the paged ROMs if they wish to claim it. If a service call is recognised then the ROM should act upon it and clear the accumulator before returning control back to the operating system. If the ROM does not wish to claim the call it should return control to the operating system with the accumulator value unchanged.

There are two sets of paged ROMs, service ROMs and language ROMs. All language ROMs should respond to paged ROM service calls and so should be service ROMs as well. BASIC is an exception to this rule and the operating system recognises it by virtue of the fact that it is a language ROM not offering a service entry.

8.1 Paged ROM header format

In order to enable the operating system to recognise ROM types and treat them accordingly, a protocol has been drawn up for a standard ROM format.

ROM offset	size	description
0	3	language entry (JMP address)
3	3	service entry (JMP address)
6	1	ROM type flag
7	1	copyright string offset pointer (=10+t+v)
8	1	version number (binary)
9	[t]	title string
9+t	1	zero byte
10+t	[v]	version string
10+t+v	1	zero byte
11+t+v	[c]	copyright string
11+t+v+c	1	zero byte
16+t+v+c	4	2nd Processor relocation address
16+t+v+c	. .	. . rest of ROM, code and data

Below is a full description of each field of the paged ROM format.

8.2 Language Entry

This should consist of a three byte JMP instruction referring to the language entry point. This code is called upon when a language is initialised, When a Tube is active the language may be copied across to the second processor and then entered, When a language is copied across the tube it may be relocated to a different address (see section 8.4 below).

If a ROM is not a language ROM this field should contain zeros.

8.3 Service Entry

This should consist of a three byte JMP instruction referring to the service entry point. This should point to code which responds to paged ROM service calls acting if and when appropriate.

If a ROM is not a service ROM this field may contain user code.

8.4 ROM Type Byte

The value of this byte gives information to the operating system about the nature of the ROM. The setting of each bit indicates a separate thing.

Bit No.	Meaning if set
0	processor/language bit
1	ditto
2	ditto
3	ditto
4	Controls Electron firm key expansions
5	Indicates that ROM has a relocation address
6	Indicates that this is a language ROM
7	Indicates that this ROM has a service entry

The first 4 bits indicate the processor type for which the code is intended, This is of importance to second processors who may get languages copied across to them. A second processor will look for the correct value of these bits before attempting to run the language. The following values have been assigned:

0	6502 BASIC
1	reserved
2	6502 code (not BASIC)
3	68000 code
8	Z80 code
9	16032 (or 32016)

If bit 5 is set this indicates that the language code in this ROM has been assembled at a different address and the ROM should be copied across the Tube to the second processor to this address. Service routines are not executed from the Tube copy.

If bit 6 is set this indicates that this is not a language ROM. This does not mean that the ROM cannot have a language entry point. If this bit is not set a language will not be considered for initialisation following a hard reset. However, if the language is entered via a service call (i.e. *<name>) a soft reset will reinitialise that language.

8.5 Copyright Offset Pointer

This is an offset value from the begining of the ROM to the zero byte preceding the copyright string, It is important that this points to a zero byte followed by '(', 'C' and ')' ASCII character values because the operating system uses this fact to determine whether a ROM physically exists in a ROM position.

8.6 Binary Version Number

This eight bit version number of the software contained in a ROM helps identify software. This byte is not used by any operating system and need not correspond to the version string.

8.7 Title String

This is a string which is printed out as the operating system enters the ROM as a language.

8.8 Version String (optional)

This should be a string identifying the release number of the software. The format of this string should be A.BB where A and B are ASCII characters of decimal digits.

On entry to a language the error pointer is set to this or if there is no version string the error pointer is directed to the copyright string.

8.9 Copyright String

This string is essential for the operating system recognition of a paged ROM (see section 8.5 above). The copyright string should always be preceded by a zero byte and start with the characters '(C)'.

8.10 The Tube Relocation address

This is the address which is used when a ROM is relocated when copying across the Tube to a second processor.

The language code should be assembled to run at that address but the service code should be assembled to run from &8000 as it will be executed within the ROM in the I/O processor.

Executing Software in Paged ROMs

It is possible to execute machine code in a paged ROM in one of three ways, via the language entry point after a reset, via the service entry point when the operating system performs a service call or via an extended vector (which is usually set up by a paged ROM in response to a service call). The following two chapters describe how the two types of paged ROMs may be implemented.