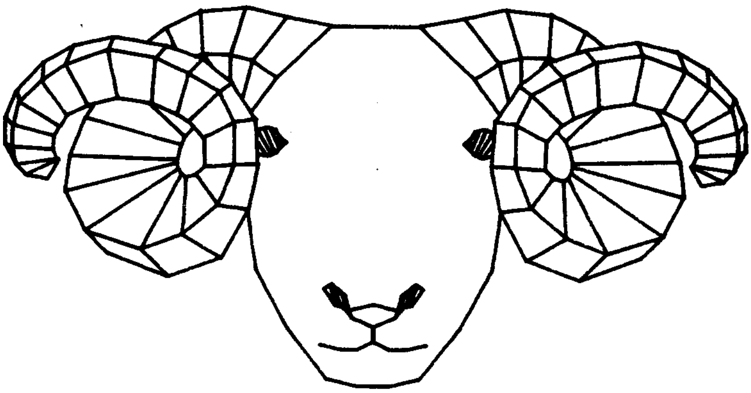
ARIES-B32

32K RAM expansion for the BBC Microcomputer Model B



Aries Computers

Introduction

How To Use This Manual

This manual is designed to be worked through from the beginning. At some points, there are examples which you will need to follow on your computer. If you attempt to jump into the middle of the text at what you think is an interesting point, you are likely to miss vital information.

Patent Notice

The principle of shadow RAM is the subject of U.K. Patent No. 2137382, held by Chris Jordan and David Barnett. This Patent Specification was published on 3 October 1984 and granted on 5 December 1985. Aries Computers Limited has a worldwide exclusive licence under this patent to produce products incorporating the shadow RAM principle for the BBC Microcomputer. It is the firm intention of all of the above parties to take all possible legal action against any infringements of this Patent.

Copyright Notice

Disclaimer

Table of Contents

Table of Contents

Basic Concepts

Before discussing specific details of the ARIES expansion system, it is necessary to understand the basic principles of shadow RAM and sideways ROM/RAM.

The 6502 processor used in the BBC Micro can access a maximum of 65536 bytes (64K) of memory directly. This limit seemed astronomical when the chip was developed, but the increasing sophistication of computer applications leaves today's user rather cramped for space. As a result a number of artificial methods have been developed whereby this 64K limit may be overcome. Of these, shadow RAM and sideways ROM/RAM are the most important.

Shadow RAM

When in an 'expensive' screen mode, giving 80-column text or colour graphics, the BBC Micro uses 20K of its RAM for the screen display (addresses in the range &3000 to &7FFF). Coupled with the demands of the operating system and disc filing system, the memory available for your programs and data falls to below 6K.

The term 'Shadow RAM' refers to the system developed for Aries Computers in 1983 (and more recently widely imitated) in which memory used for the video display of the BBC Micro, and thus unavailable for programs and data, is replaced by a shadow bank of extra RAM. This shadow RAM occupies the same processor addresses as the video RAM, with the selection between the two banks being performed by special control software invisible to normal programs. The control software ensures that information to be displayed is routed to the video RAM, whilst programs and data are routed to shadow RAM.

Shadow RAM makes around 26K of memory available for your programs and data at all times, regardless of screen mode. (Note that this is not a significant improvement in mode 7, as this display takes very little memory.)

Sideways ROM

The sideways ROM system on the BBC Micro enables a number of pieces of software in ROM to share the same area of the 6502 processor's address space. This means that up to 256K of ROM software takes up just 16K of the 64K that the 6502 can address (addresses in the range &8000 to &BFFF). The 256K is arranged in banks of 16K which appear to be ' sideways' from each other in the address space.

Since only one bank of sideways ROM can be addressed by the processor at any instant, the appropriate bank is selected automatically by the operating system.

Sideways RAM

Sideways RAM is an extension to the sideways ROM system, providing one or more banks of RAM in place of sideways ROM. This RAM behaves exactly like the equivalent sideways ROM sockets, except that, whilst data in a ROM chip needs to be physically plugged into a sideways ROM socket, data can be written into a sideways RAM bank by software.

In addition to the potential for using sideways RAM for data storage or as a printer (or other) buffer, you can load your own sideways ROM software from disc. This allows you to have a vast library of sideways ROMs without running out of sockets.

You should note that copying commercial ROM software to disc for this purpose without the authors' permission is strictly forbidden by copyright law. Many software suppliers have special licensing arrangements for users such as schools who wish to use multiple copies of a single ROM in this way.

Overview of the ARIES Expansion System

Since the introduction of the ARIES-B20 shadow RAM board in 1983, a number of products based on the same principle have appeared, of which the most notable is the BBC Micro Model B+.

Based on two years' and several thousand users' experience of ARIES-B20, Aries Computers has developed a sophisticated 'second generation' expansion system for the BBC Microcomputer model B, consisting of the ARIES-B32 RAM board and the ARIES-B12 sideways ROM board.

ARIES-B32 is a single plug-in board, giving you an extra 32K of RAM plus a 16K sideways ROM socket. You can configure the extra 32K of RAM in a variety of ways, using simple commands from the keyboard or within programs:

- as 20K of shadow RAM plus 12K of sideways RAM
- as 16K of shadow RAM, plus 16K of sideways RAM
- as two 16K banks of sideways RAM.

The sideways ROM socket on ARIES-B32 is used to contain the ARIES-B32 ROM. In addition to controlling the allocation and switching of shadow RAM, this ROM contains a variety of powerful utility commands.

ARIES-B12 is a two-part sideways ROM board, giving you 12 sockets for software in sideways ROM plus the option to fit 16K of low-cost sideways RAM. The two-part design allows you to fit ARIES-B12 either inside or outside your BBC Micro's case.

Together, ARIES-B32 and ARIES-B12 expand your BBC Micro to 80K of RAM and a full 16 sideways ROM/RAM slots - a better specification than the BBC Micro model B+.

Fitting ARIES-B32 - Part 1

Introduction

You should have no difficulty in fitting your ARIES-B32, if you follow the instructions below carefully. Please read the instructions all the way through before starting.

If you are fitting ARIES-B12 at the same time as ARIES-B32, fit and test ARIES-B32 first. You should not fit ARIES-B12 until you have ARIES-B32 working correctly.

When handling your ARIES-B32 take great care not to bend any of the pins of the 40-way connector on the bottom of the board. If you need to remove ARIES-B32 from your BBC Micro for any reason, follow the instructions in the section headed 'Having Problems?', below.

Tools Required

You will need:

1) A medium cross-point screwdriver

2) A medium flat-bladed screwdriver

3) A pair of fine-nosed pliers

Parts Supplied

You should have:

1) A circuit board with a ROM chip fitted to it

2) A small plastic support pillar

Removing Chips

In order to fit your ARIES-B32, you will need to remove the processor chip from its socket on your BBC Micro. If you wish to rearrange the sideways ROMs in your computer you will need to remove them from their sockets. To avoid the risk of damaging these chips or their sockets, you should use the method described below.

Take a flat-bladed screwdriver or similar tool and remove the chip by prising it up a little at a time at each end. Put the tip of the screwdriver between the chip and the plastic body of the socket and try not to lever against the surrounding chips. You can use a pencil or similar small object laid flat on the circuit board to lever against if necessary. Take care not to bend the pins on the chip, especially when it is about to come free of its socket.

A similar technique will also work for ARIES-B32 and ARIES-B12, should you need to remove them.

Opening the BBC Micro

You will need to open the case of your BBC Micro to fit ARIES-B32. To do this, remove the four main case screws (usually marked 'FIX'). Two of these are on the back of the computer and the other two are on the underside. When you have removed these four screws you should be able to lift the top of the computer off. Put the top and the screws to one side.

Having removed the top of the computer, undo the bolts holding the keyboard to the computer. You will find these at the sides of the keyboard. Some BBC Micros have three bolts, others have only two. Put the bolts aside with the top of the computer and the other screws. You need not disconnect the keyboard completely; simply slide it clear of the main circuit board (see figure 1).

Fitting the Circuit Board

First, take the small nylon support pillar and clip it into the large hole in ARIES-B32, on the bottom of the circuit board.

Identify the 6502 processor chip (see figure 1). The number '6502' should be marked on the top of the chip. Following the instructions headed 'Removing Chips', remove the 6502 chip from its socket.

Take the 6502 chip and 'fit it into the 40-pin socket on the ARIES-B32. Make sure that you fit the chip the right way round (see figure 2). The notch or dot on one end of the chip should match up with the notches or bumps on the other chips on ARIES-B32. Check that all the pins have entered the socket correctly and are not bent underneath the chip.

Carefully fit ARIES-B32 into the socket from which you removed the 6502 chip (see figure 3). Make absolutely sure that the 40-pin connector on the bottom of ARIES-B32 is lined up exactly with the 40-way socket in the BBC Micro.

The ROM software for ARIES-B32 is supplied already fitted to the sideways ROM socket on ARIES-B32. Normally, you should not need to move it (though you are quite free to do so if you wish).

Figure 1

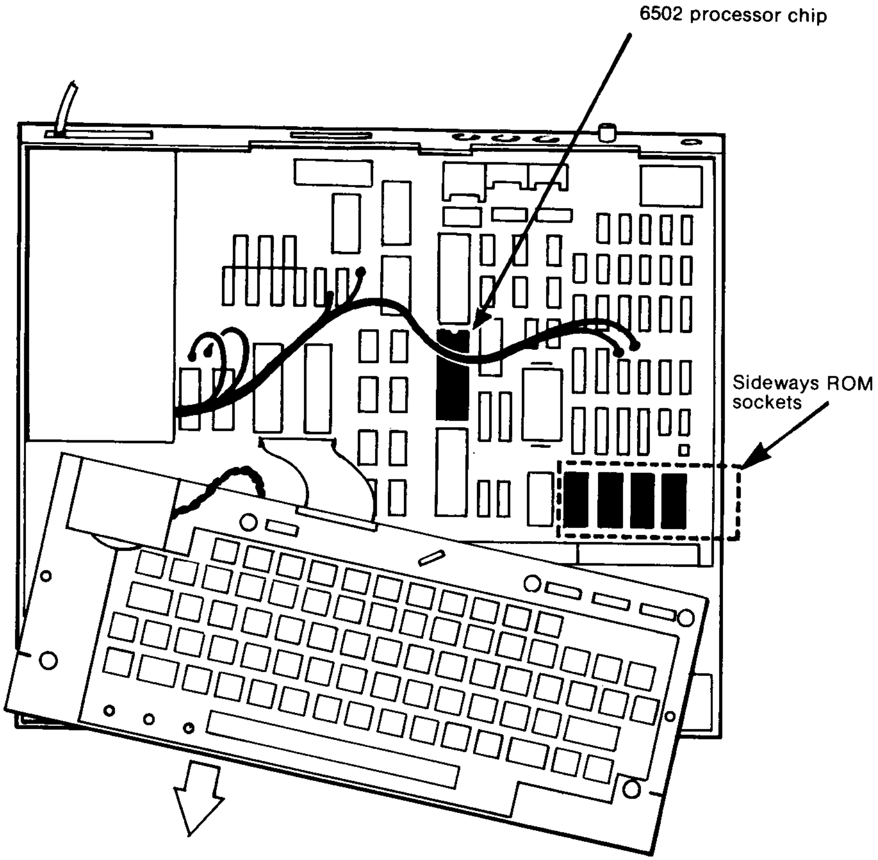


Figure 2

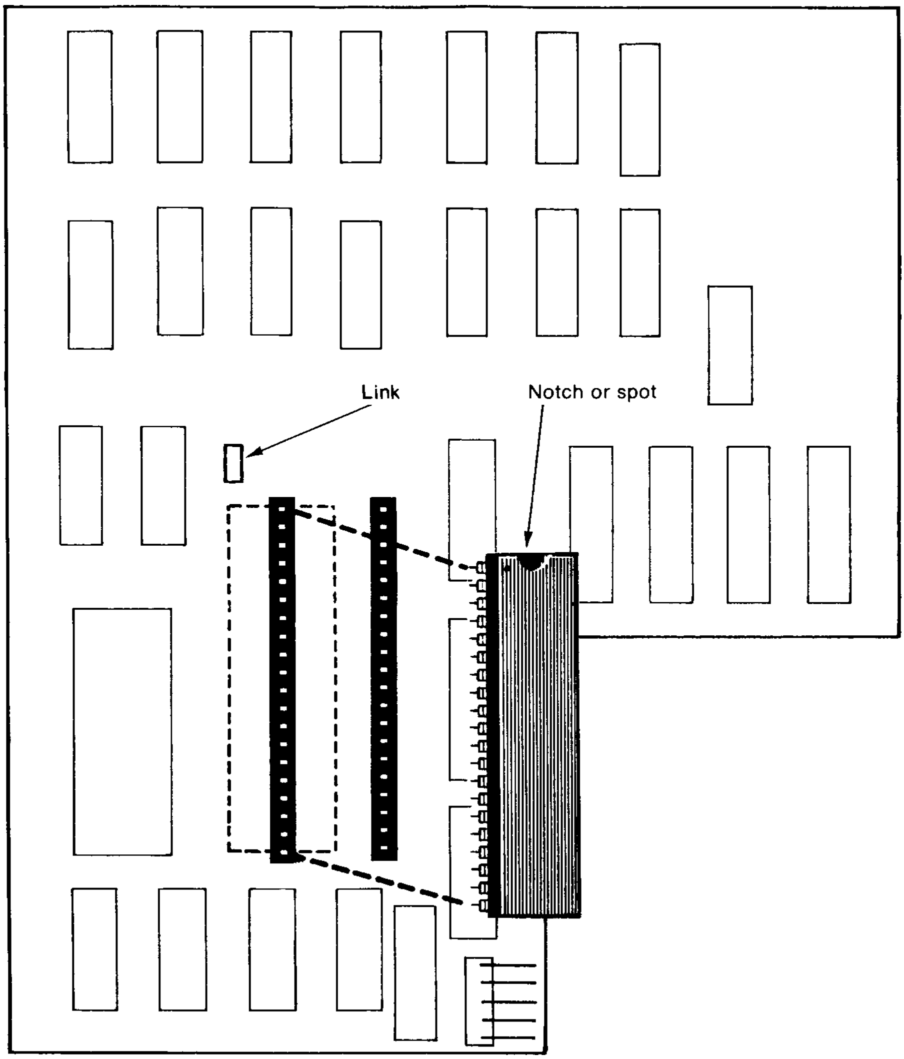
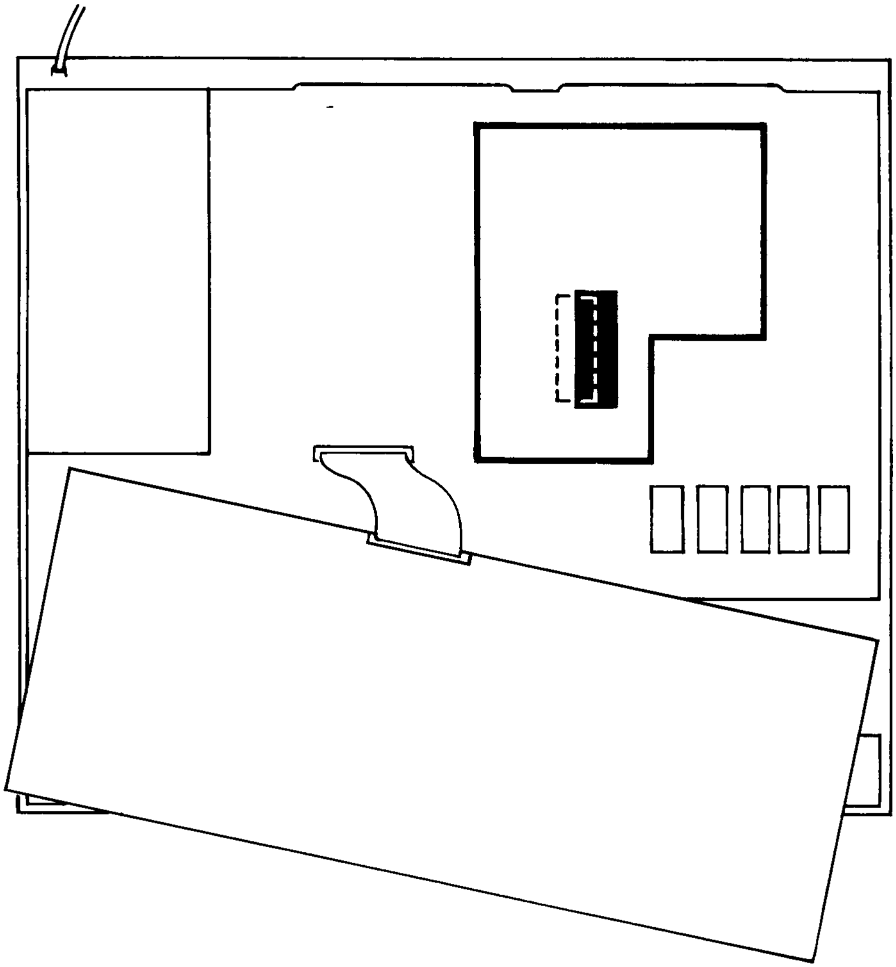


Figure 3



ARIES-B32 has the side-effect of rearranging the priority of your BBC Micro's standard sideways ROM sockets. This means that when you turn the computer on, you may find that it has started up in a different program from before (for example VIEW instead of BASIC). A detailed explanation of this effect, and what to do about it, is given later on in this manual. For now, you should remove all the sideways ROMs except BASIC and DFS. Follow the instructions given above, under the heading 'Removing Chips'. Remember that the socket immediately to the left of the four sideways ROM sockets contains the operating system ROM. Do not remove this ROM.

You are now ready to test your ARIES-B32!

Testing ARIES—B32

Reposition the keyboard in its proper place, but do not replace the bolts yet. Check that the keyboard is not touching any metal parts of the computer. Make sure that there are no tools lying inside the computer. Replace the top of your computer, but do not replace the fixing screws. Connect the computer to a television or monitor and switch the television or monitor on.

Now turn on your BBC Micro. If you do not hear the familiar 'beep' and see a normal screen display, turn the computer off instantly and refer to the section headed 'Having Problems?', below.

If your BBC Micro appeared to start up normally, type:

```
*XTEST{RETURN}
```

(Note that, in this manual, a word shown inside curly brackets thus: '{RETURN}', represents the single key marked 'RETURN'.)

ARIES-B32 will perform a thorough check of itself and print a series of messages to tell you the results. NOTE: the test may take several minutes to complete: this is perfectly normal. The final message should be:

```
  Test OK. Press BREAK to continue:
```

If the test finishes with this message, congratulations! You now have a working ARIES-B32 installation; go on to the tutorial section below.

If the test finishes with some other message, turn the computer off and refer to the section headed 'Having Problems?'.

Having Problems?

NOTE: if you need to remove ARIES-B32 from your BBC Micro, you should remove the board from the processor socket by lifting it gently, a little at a time, alternately at the front and back until it comes free of the socket. Continue to handle ARIES-B32 carefully when it is out of the socket. If you are too rough, you risk bending or breaking one of the pins. To remove the 6502 chip from the socket on ARIES-032, follow the instructions headed 'Removing Chips' above; take particular care not to lever against the circuit board or components. Replace the 6502 chip in the processor socket in your computer with the end marked with a notch or dot pointing to the back of the computer.

If your computer appears not to function correctly when you fit ARIES-B32 to it, check the following points:

- Is the computer plugged in to the mains?
- Is the television or monitor connected to the computer, plugged in and switched on?
- Has the keyboard connector become disconnected?
- Is the 6502 chip fitted the right way round?
- Are there any bent pins on the 6502 chip?
- Is ARIES-B32 lined up correctly in the 40-way socket?
- Does your BBC Micro have version 1.2 or later of the MOS fitted? (If you do not know how to check this, consult your dealer who will be able to supply the correct version of the MOS if necessary.)
  Has your BBC Micro been upgraded to model 'B' correctly?
- Is some other piece of hardware or ROM software interfering with ARIES-B32? (Try removing all 'suspects' and replacing them one at a time until the problem appears.)

If you still cannot see any reason why the computer does not work, consult your supplier.

Tutorial Guide to the ARIES System

Introduction

This section of this manual attempts to give you a fuller understanding of the ARIES-B32 and ARIES-B12 by means of a series of explanations and examples. It is by no means essential that you understand everything straight away; indeed, there are many things you may never need to know. Rather, you should use this tutorial, and the advanced tutorial that follows it, to gain an overall impression of the system and refer to the detailed specifications in the technical reference section as and when you need to.

This tutorial is designed to be worked through from the beginning, so you should resist the temptation to jump in at an interesting-looking point. In particular, some of the example commands will not work in the way described if you have not worked through all of the previous examples.

To start, turn your computer off, then on again, to ensure that it is in a well-defined state. For the purposes of this tutorial we have assumed that your computer starts up in BASIC in screen mode 7 (teletext mode).

System Status Information

Before you attempt to use ARIES-B32, you need to know how to find out what is going on in your computer. ARIES-B32 provides all of the necessary information by means of the command *XSTATUS. Using *XSTATUS, you can experiment with ARIES-B32 commands for yourself, and see their results.

Try typing:

```
*XSTATUS{RETURN}
```

You should see a display something like:

```
Current shadow RAM: 20K
Workspace page: dynamic
Start-up link: ON
Program/data RAM: 26112 bytes
From: &1A00 To: &8000
```

(Don't worry if the numbers in the last two lines aren't exactly the same as our example.)

As you work through this tutorial, you will recognise the various parts of the *XSTATUS display and understand their importance.

Shadow RAM

Take a look at the *XSTATUS display. The first line tells you that you have 20K of shadow RAM active. Note the amount of program/data RAM available (the fourth line of the *XSTATUS display). This is the space which is available for your programs and variables, your text in word-processors, your spreadsheet models, etc.

Now switch into a different screen mode:

```
MODE 1{RETURN}
```

followed by:

```
*XSTATUS{RETURN}
```

Notice that, even though you have changed to a screen mode which uses 20K of RAM, you still have the same amount of program/data RAM.

For straightforward use, you now know as much about shadow RAM as you need to! Unfortunately, however, there are some programs which, because of the way in which they are written, do not work with shadow RAM systems (the most obvious examples are games). To enable you to run such programs conveniently, ARIES-B32 is provided with a command which disables shadow RAM and returns your computer as nearly as possible to its original state.

Type:

```
*XOFF{RETURN}
```

you will see the message:

```
B32: Press BREAK to continue:
```

(All of ARIES-B32's messages start 'B32: ' to enable you to distinguish them from similar messages produced by other software.) Because you are making a significant alteration to the configuration of your BBC Micro, ARIES-B32 is forcing you to press {BREAK} to give your computer the chance to adjust to the change. Your computer will not respond to any keys until you press {BREAK}, so do so now.

Use *XSTATUS to see what changes have occurred:

```
*XSTATUS{RETURN}
```

The *XSTATUS display should look like this:

```
Current shadow RAM: OK
Workspace page: none
Start-up link: ON
Program/data RAM: 25344 bytes
From: &1900 To: &7C00
```

Naturally, you now have no shadow RAM. Your program/data RAM is only reduced by 768 bytes because you are back in screen mode 7, which only uses 1K of RAM (1024 bytes), and ARIES-B32 has given up 256 bytes of RAM which it was using for workspace (notice the second line of the *XSTATUS display).

Now try a 'greedy' screen mode again:

```
MODE 1{RETURN}
```

The *XSTATUS display should look like this:

```
Current shadow RAM: OK
Workspace page: none
Start-up link: ON
Program/data RAM: 5888 bytes
From: &1900 To: &3000
```

Without shadow RAM you have 20244 bytes less program/data RAM available, but you are able to run almost any software written for the BBC Micro just as if ARIES-B32 were not fitted.

To get shadow RAM back, type:

```
*XON{RETURN}
```

Once more, you will see:

```
B32: Press BREAK to continue:
```

Press BREAK and your computer will be back in the state it started from (you might like to check this with *XSTATUS).

At the moment, whenever you turn your computer on, it will start up with 20K of shadow RAM active. When you press {CTRL}{BREAK} (hold {CTRL} down while you press and release {BREAK}), it will be reset to the 20K shadow RAM state. When you press {BREAK} the current configuration will be retained.

To demonstrate this, switch off shadow RAM with *XOFF. Use *XSTATUS to check that this has worked. Now press {BREAK} again. *XSTATUS shows that the configuration is unchanged. Press {CTRL}{BREAK} and you will see that your computer has returned to the 20K shadow RAM state.

If you would find it more convenient to have your computer start up with shadow RAM off, and turn shadow RAM off when you press {CTRL}{BREAK}, you should look in the technical reference section of this manual under the heading 'Start-Up Link'.

Sideways ROM/RAM

Introduction

At the simplest level, there is very little you need to know about sideways ROM/RAM, except which sockets to plug your sideways ROMs into. ARIES-B32 has a variety of commands to control and manipulate sideways ROM/RAM systems, but the majority of these are provided for sophisticated uses.

Sideways ROMs

Sideways ROMs for the BBC Micro fall into three broad groups:

1) Languages

For the purposes of describing sideways ROMs, 'languages' includes not only programming languages, such as BASIC and LOGO, but also programs such as the VIEW word processor and ViewSheet spreadsheet.

Only one language can be active at any one time.

2) Filing Systems

This group includes the disc filing system and the network filing system. (The cassette filing system is built into the operating system and is not a sideways ROM.)

Only one filing system can be active at any one time.

3) Services

'Services' are ROMs which extend the operating system, such as the ARIES-B32 ROM, or the Acornsoft Graphics Extension ROM.

A number of services may be active at any one time.

In practice, some ROMs are a combination of the above. For example, disc filing system ROMs also contain utility commands, which makes them services as well as filing systems.

Priority and Choosing Sockets

When you switch your computer on, or press {CTRL}{BREAK}, the operating system has to choose which language and which filing system should be active. This decision is made by giving the physical ROM sockets a priority order, and taking the highest priority language and filing system.

The BBC Micro's operating system can support up to sixteen sideways ROMs. On the unexpanded model B, there are only four sockets, numbered 12 to 15 from left to right. On ARIES-B12, there are twelve ROM sockets, numbered 1 to 12, and two RAM sockets which behave like a single ROM socket, numbered 0L and 0H (for socket 0, low half and socket 0, high half). On ARIES-B32, there is one ROM socket, numbered 15, and two banks which may contain RAM (depending on how much RAM has been used for shadow RAM), numbered 13 and 14.

In the case of a system with both ARIES-B32 and ARIES-B12, all sixteen sockets are accounted for. If you do not have ARIES-B12 then three of the BBC Micro's sockets might seem to have the same numbers as the banks provided by ARIES-B32. ARIES-B32 resolves this for you, by renumbering the BBC Micro's sockets. This gives the four sockets on your BBC Micro the numbering 12, 9, 10, 11 from left to right.

In each case, the highest numbered socket has the highest priority.

This system of priority means that you must choose carefully which ROMs you put into which sockets, in order that your computer will switch on in your preferred state. If, for instance, you wish to start up in BASIC, and you also possess LOGO and VIEW, you should ensure that BASIC is fitted in a higher numbered socket than either of the others. Similarly, if you possess more than one filing system ROM, the one in the highest numbered socket will take priority.

After the highest priority one of each, the ordering of languages and filing systems should make no difference. In a perfect world, the ordering of all of your service ROMs would make no difference. As it is, however, you may find that certain service ROMs have quite precise requirements as to positioning. The manuals for such ROMs should give you the information you need in order to decide which sockets to choose for them.

Sideways ROM/RAM Status

To continue with our practical demonstration, type:

```
*XSTATUS S{RETURN}
```

You should see something like:

```
 9: ROM 8K DFS 0.90
12: ROM 16K BASIC
14: RAM 12K Unused
15: ROM 16K ARIES-B32 1.00
RWRITE OFF
```

(obviously, this may differ for your particular system, according to precisely which ROMs you have).

On each line, the number in the left-most column is the sideways bank number, the next column tells you whether there is ROM or RAM in that bank, the third column tells you what size the bank is, and the final column tells you what the bank contains. Banks containing neither RAM nor a valid sideways ROM are not displayed.

For now, the last line of the *XSTATUS sideways display is not of interest to us.

Extended Buffers

ARIES-B32 provides a set of utility commands to allow you to use sideways RAM to extend any one of the operating system's buffers. The commonest use of this facility is to provide an extended printer buffer which permits you to continue working while a long document is being printed. If you use a terminal program you will find that the reliability and performance of your system will be improved by extending the RS423 input buffer.

Type:

```
*XSTATUS S{RETURN}
```

At present, you should see that sideways bank 14 consists of 12K of unused RAM. Type:

```
*BUFFER PRINTER 14{RETURN}
```

This instructs ARIES-B32 to create an extended printer buffer in sideways bank 14. Type:

```
*XSTATUS S{RETURN}
```

You should see something like:

```
 9: ROM 8K DFS 0.90
12: ROM 16K BASIC
14: RAM 12K Buffer PRINTER
15: ROM 16K ARIES-B32 0.13
RWRITE OFF
```

Now try:

```
*XSTATUS B{RETURN}
```

You will see:

```
Buffer: PRINTER
Capacity: 11788 bytes
Contents:       0 bytes
Purge: OFF
```

At the moment, the third line of the *XSTATUS buffer display tells us that the extended buffer is empty. To put some characters into the buffer, first ensure that your printer (if you have one) is switched off, then type {CTRL}{B} (press and release {B} while holding {CTRL} down). Type:

```
  REM This is some data for the buffer{RETURN}
```

Type {CTRL}{C} then:

```
   *XSTATUS B{RETURN}
```

You will see that there are now some characters in the extended printer buffer:

```
  Buffer: PRINTER
  Capacity: 11788 bytes
  Contents: 37 bytes
  PURGE OFF
```

To remove the extended buffer, try typing:

```
  *BUFFER OFF{RETURN}
```

You will see an error message which tells you that you cannot change the state of a buffer which contains data. To get rid of the contents of the extended buffer type:

```
  *PURGE{RETURN}
```

*XSTATUS will show that the extended buffer is empty once more. You can now type:

```
  *BUFFER OFF{RETURN}
```

This time the buffer will be removed successfully.

Fitting ARIES-B32 - Part 2

Fitting ARIES—B12

If you are going to fit ARIES-B12 to your computer, now is the time to do so. Full instructions are given in the ARIES-B12 manual.

Fitting Sideways ROMs

The information given above in the Tutorial Guide should have enabled you to decide on an ordering for your sideways ROMs. Fit the ROMs into the ROM sockets according to this plan. (If you have ARIES-B12, refer to the ARIES-B12 manual, otherwise see figure X).

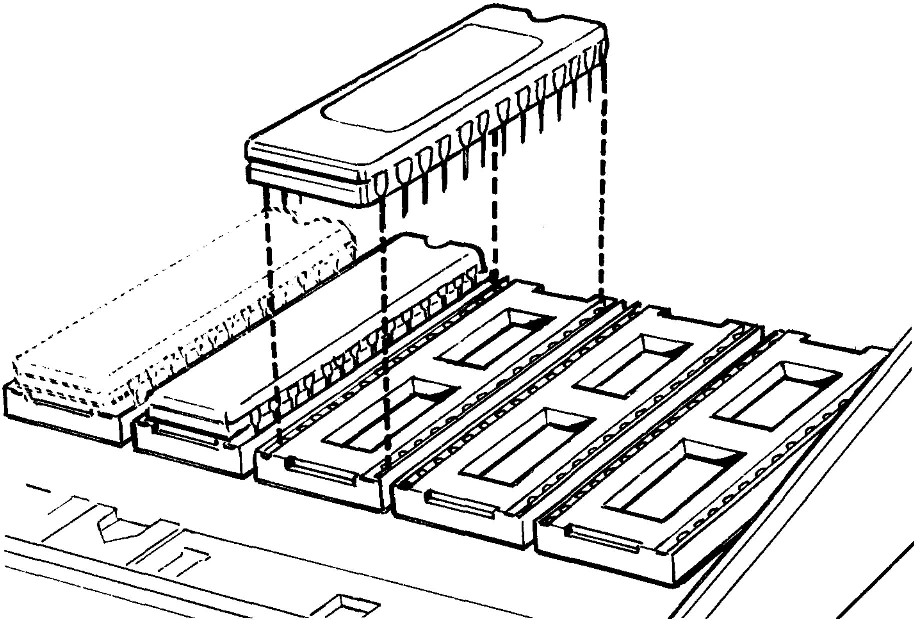
Final Testing

Before reassembling your computer, switch it on once more and check that it is still working correctly. Switch the computer off again.

Reassembling the BBC Micro

Replace the bolts securing the keyboard to the computer (the washers and nuts go inside the computer). Replace the top of the computer and secure it with the four large screws. Switch your BBC Micro on again to check that it is still working.

Figure 4



Advanced Tutorial on the ARIES System

Shadow RAM

One of the unique features of ARIES-B32 is its ability to split its 32K of RAM into 16K of shadow RAM and 16K of sideways RAM. So far we have seen ARIES-B32 configured as 20K of shadow RAM and 12K of sideways RAM (when you switched your computer on) and OK of shadow RAM and 32K of sideways RAM (after *XOFF). Type:

```
*XON 16{RETURN}
```

Don't press {BREAK}! Notice that no message appears telling you to press {BREAK}. This is because the change from 20K of shadow RAM to 16K of shadow RAM is much less drastic than that from 20K of shadow RAM to none. That doesn't mean that the change has actually occurred yet, though. Take a look at the *XSTATUS display:

```
*XSTATUS{RETURN}
```

You should see:

```
Current shadow RAM: 20K
Pending shadow RAM: 16K
Workspace page: dynamic
Start-up state: ON
Program/data RAM: 26112 bytes
From: &1A00 To: &8000
```

The second line of the display is new. It tells you that when you next change screen mode (or press {BREAK}) your computer will be reconfigured with just 16K of shadow RAM. Now change screen mode (setting the screen mode to be the same as at present counts as a change for this purpose):

```
MODE 7{RETURN}
```

The *XSTATUS display shows that the change has now happened.

Note that, so far, the amount of program/data RAM available has not changed. This is because 16K of shadow RAM is enough to replace the entire video display RAM in screen modes 3 to 7. What happens when you move to a screen mode which uses 20K of video RAM? Type:

```
MODE 1{RETURN}
```

Screen mode 1 uses 20K of RAM and the *XSTATUS display shows that the 4K excess over the 16K of shadow RAM has been taken away from the available program/data RAM:

```
Current shadow RAM: 16K
Workspace page: dynamic
Start-up state: ON
Program/data RAM: 22016 bytes
From: &1A00 To: &7000
```

Finally, to return once more to the 20K shadow RAM state, type:

```
  *XON{RETURN}
  MODE 7{RETURN}
```

Verify that this has worked, using *XSTATUS.

Switching Off Shadow RAM Temporarily

It is often desirable to be able to perform an operating system command with video RAM switched in instead of program RAM. Examples of this are saving and loading screens and dumping screens to printers. ARIES-B32 provides a facility to allow this. Type:

```
  MODE 1{RETURN}
  *HELP ARIES{RETURN}
  *X*SAVE SCREEN 3000 8000{RETURN}
```

The *X* command causes ARIES-B32 to switch off shadow RAM and then pass the remainder of the command-line to the operating system. On completion of the command, shadow RAM is switched on once more. Type:

```
  CLS{RETURN}
  *X*LOAD SCREEN{RETURN}
```

to check that all of this has been successful.

A word of warning! You must not use *X* to enter a language ROM with shadow RAM switched off. This might appear to work, but will tend to crash your computer unpredictably after a while. Try typing:

```
  *X*BASIC{RETURN}
```

*XSTATUS shows that the top of memory has moved down as if shadow RAM were off, but 20K of shadow RAM is still allocated. This is not a stable state! Press {ESCAPE}. Use *XSTATUS to see what has happened.

Switching Off Shadow RAM Without Pressing {BREAK}

So far, we have seen how to switch shadow RAM on and off in a very thorough way, pressing {BREAK} to re-configure the computer completely. We have also seen how the change between 20K of shadow RAM and 16K of shadow RAM is much less drastic.

There is a way to change from 20K of shadow RAM to none without pressing {BREAK}. This method has another advantage. Type:

```
  *XSTATUS{RETURN}
```

Write down the value given for the start of program/data RAM in the last line of the *XSTATUS display. Type in a small BASIC program:

```
10 PRINT "Hello World!"{RETURN}
RUN{RETURN}
```

Now switch off shadow RAM using *XOFF. Take a look at the *XSTATUS display. You will see that the start of program/data RAM is now &100 lower than it was when shadow RAM was active. Try and retrieve your BASIC program by typing:

```
OLD{RETURN}
```

You will see the message:

```
Bad Program
```

This tells you that your program is not there!

What has happened is this: ARIES-B32 needs 256 bytes (&100 bytes in hexadecimal notation) of workspace to control shadow RAM. This workspace is located below your programs and data in the sideways ROM workspace area. When you turned shadow RAM off with *XOFF, ARIES-B32 gave up its workspace, so the RAM available for your programs and data began &100 bytes lower down memory. Your simple BASIC program has not been lost, but there are now 256 bytes of rubbish before its start, which prevents BASIC from finding it.

Use *XON to recover shadow RAM. This also causes ARIES-B32 to take its workspace once more. Try again to recover your BASIC program. You will find that it is alive and well!

In order to avoid the need to press {BREAK} and the problem of mislaying your programs and data, a special version of *XOFF is provided. Type:

```
*XOFF 2{RETURN}
```

The first thing you will notice is that you have not been forced to press {BREAK}. Try *XSTATUS. This time you should see:

```
Current shadow RAM: 20K
Pending shadow RAM: 0K
Workspace page: dynamic
Start-up state: ON
Program/data RAM: 26112 bytes
From: &1A00 To: &8000
```

In just the same way as in our example for *XON 16, the second line of the *XSTATUS display tells you what the configuration of your computer will be after you next change screen mode or press {BREAM. Type:

```
MODE 7{RETURN}
```

*XSTATUS will show:

```
   Current shadow RAM: 0K
   Workspace page: dynamic
   Start-up state: ON
   Program/data RAM: 25088 bytes
   From: &1A00 To: &7C00
```

Notice that ARIES-B32 still has its workspace. You have not had to press {BREAK} and your program has not been affected at all. In fact, you can use *XOFF 2 within programs to switch off shadow RAM temporarily, using *XON to switch it back on again.

You might like to consider the current configuration to be 'OK of shadow RAM', rather than 'shadow RAM switched off'. It might even be argued that *XOFF 2 should be renamed *XON 0. Try using *XON, *XON 16, and *XOFF 2 to move between the three possible configurations (use *XSTATUS to observe the results).

You might expect that machine-code programs which need shadow RAM to be switched off would run correctly following *XOFF 2. Beware! This is not necessarily the case. Many such programs must run at a fixed address in RAM, often overlapping the memory used by ROMs such as ARIES-B32 for workspace.

When you used *XOFF 2 to switch off shadow RAM, ARIES-B32 retained its workspace. If you overwrite that workspace with a program, you will crash your computer. What is needed is a combination of *XOFF and *XOFF 2. ARIES-B32 does provide such a command: *XOFF 1.

Try using *XOFF 1 instead of *XOFF 2. You probably won't be able to see any difference whatever. ARIES-B32 still appears to retain its workspace. The difference is very subtle. When you use *XOFF 2, ARIES-B32 continues to use its workspace in much the same way as it did when shadow RAM was switched on. When you use *XOFF 1, ARIES-B32 tidies up its workspace and stops using it (remember, this doesn't happen until the next mode change or {BREAK}), so it may be overwritten in safety.

You are probably wondering, at this point, why there is a need for *XOFF 2 at all. The answer is that, as part of its tidying up process when you use *XOFF 1, ARIES-B32 may well tidy up and switch off other ROMs in your system. This makes *XOFF 1 rather unsatisfactory for normal use.

As a general rule, you should use *XOFF 2 wherever possible, especially within programs. *XOFF 1 is provided for extreme cases, such as games software, if *XOFF 2 has been found not to work.

Sideways ROM/RAM

To start off with, ensure that your system is configured with 20K of shadow RAM by using *XON. Type:

```
*XSTATUS X S{RETURN}
```

You will see something like:

```
Current shadow RAM: 20K
Workspace page: dynamic
Start-up state: ON
Program/data RAM: 26112
bytes From &1A00 To: &8000
 9: ROM 8K DFS 0.90
12: ROM 16K BASIC
14: RAM 12K Unused
15: ROM 16K ARIES-B32 1.00
RWRITE OFF
```

From this you can see that the 32K of RAM provided by ARIES-B32 is configured as 20K of shadow RAM and 12K of sideways RAM in bank 14.

In order for us to be able to experiment further with sideways RAM, we need a sideways ROM image. To avoid infringing copyright, we will generate a tiny ROM of our own. Enter the following BASIC program:

```
1000 REM Tiny ROM generator
1010 *RWRITE 14
1020 base% = &8000
1030 title$ = "MYROM8"
1040 version% = 1
1050 version$ = "0.01"
1060 copyright$ = "(C) 1985 Mysoft"
1070 copyright% = base%
1080 FOR pass% = 0 TO 3 STEP 3
1090 P% = base%
1100[OPT pass%
1110 JMP language
1120 JMP service
1130]
1140 ?P% = &81
1150 P%?1 = copyright% - base%
1160 P%?2 = 123
1170 P% = P% + 3
1180 $P% = title$
1190 P% = P% + LEN (title$)
1200 ?P% = 0
1210 P% = P% + 1
1220 $P% = version$
1230 P% = P% + LEN (version$)
1240 ?P% = 0
1250 copyright% = P%
1260 P% = P% + 1
1270 $P% = copyright$
1280 P% = P% + LEN (copyright$)
1290 ?P% = 0
1300 P% = P% + 1
1310[
1320.language
1330 RTS
1340.service
1350 RTS
1360]
1370 NEXT pass%
1380 *RWRITE OFF
```

Now save this program, then run it. Type:

```
*XSTATUS S{RETURN}
```

The *XSTATUS display should now read something like:

```
9: ROM 8K DFS 0.90
12: ROM 16K BASIC
14: RAM 8K (MYROM8 0.01)
        4K Unused
15: ROM 16K ARIES-B32 1.00
RWRITE OFF
```

The brackets round the entry for our ROM are there to tell you that the ROM has not been recognised by the operating system. This will only happen when you next press {BREAM. Do so now, then use *XSTATUS to check that our ROM has been recognised.

We need to save the image of our tiny ROM. Type:

```
*XMOVE S14.8000 A000 P.3000{RETURN}
*SAVE MYROM8 3000 5000{RETURN}
```

This has saved an 8K image of our ROM. We will need a 16K version as well, so change the title of the ROM in the program to 'MYROM16' and run the program again. Type:

```
*XMOVE 514.8000 C000 P.3000{RETURN}
*SAVE MYROM16 3000 7000{RETURN}
```

Note that the working part of our ROM is only a few bytes long. The rest of it is random rubbish. This does not matter at all. If you check with *XSTATUS S, you will find that MYROM16 still appears as an 8K ROM image at present, because only the first few bytes are present in the sideways RAM bank.

Before going on, we must remove our ROM image from bank 14. Type:

```
*RKILL MYROM16{RETURN}
```

You will be told to press {BREAK}. When you have done so, use *XSTATUS to check that the image has been removed. Now we will try loading our ROM image into sideways RAM once more. Type:

```
*RLOAD MYROM16 14{RETURN}
```

You will see an error message telling you that there is not enough sideways RAM in bank 14 to load the 16K ROM image. Use *XON 16 to reconfigure the ARIES-B32 RAM so that there is 16K of shadow RAM and 16K of sideways RAM (don't forget to change mode or press {BREAK} after *XON 16).

Try *RLOAD again. This time you will see a message reminding you that the operating system will not recognise the new ROM until you press {BREAK}. Note that you are not forced to press {BREAK} at this point. The *XSTATUS display should look something like this:

```
  9: ROM 8K DFS 0.90
  12: ROM 16K BASIC
  14: RAM 16K (MYROM16 0.01)
  15: ROM 16K ARIES-B32 1.00
  RWRITE OFF
```

Since 16K of ARIES-B32's RAM is now in use as sideways RAM, it should be impossible to reconfigure your system to get 20K of shadow RAM. Try typing:

```
  *XON{RETURN}
```

No error message appeared. *XSTATUS tells us:

```
  Current shadow RAM: 16K
  Workspace page: dynamic
  Start-up state: ON
  Program/data RAM: 26112 bytes
  From &1A00 To: &8000
```

This does not mean that *XON was ignored. In fact, when you type *XON with no argument, you are asking for as much shadow RAM as possible, either 16K or 20K. In this case, 16K is all that is available, so 16K is what you got. You can insist on 20K of shadow RAM by typing:

```
  *XON 20{RETURN}
```

This time you will see an error message telling you that the extra RAM needed is already in use elsewhere. In general, ARIES-B32 will not let you take RAM which is already in use. There is one exception. Type:

```
  *RLOAD MYROM8 14{RETURN}
```

You will see a message telling you that the previous contents of sideways RAM bank 14 have been overwritten. In this case you are forced to press {BREAK} to give your computer the chance to adjust to the change. Once you have done so, *XSTATUS will show something like:

```
  9: ROM 8K DFS 0.90
  12: ROM 16K BASIC
  14: RAM 8K MYROM8 0.01
             8K Unused
  15: ROM 16K ARIES-B32 1.00
  RWRITE OFF
```

Remove our ROM image with *RKILL. Switch shadow RAM off completely with *XOFF 2. Use *RLOAD to load MYROM8 into sideways RAM bank 13. When you have finished, the *XSTATUS display will look something like:

```
 9: ROM 8K DFS 0.90
12: ROM 16K BASIC
13: RAM 8K MYROM8 0.01
        8KUnused
14: RAM 16K Unused
15: ROM 16K ARIES-B32 1.00
RWRITE OFF
```

Try typing:

```
 *XON{RETURN}
```

Even though there is 24K of ARIES-B32's RAM unused, you are not allowed to reconfigure it. This is because ARIES-B32 always tries to take RAM from sideways RAM bank 13 when it needs shadow RAM.

*RKILL may be used to disable temporarily sideways ROMs as well as to remove ROM images from sideways RAM. Type:

```
*RKILL MYROM8 BASIC{RETURN}
```

Press [BREAK}. Instead of the normal BASIC prompt, you will see a star, '*'. Because we have disabled the language we were using, ARIES-B32 has provided its own tiny language. This language, called 'OSCLI', only allows you to type operating system commands (including those added by sideways ROMs, of course). Type:

```
XSTATUS S{RETURN}
```

Note that you did not need to type a '*' before XSTATUS. You will see that BASIC is disabled. Press [BREAK). Even though BASIC has been re-activated, you are still in OSCLI. To return to BASIC, type:

```
BASIC{RETURN}
```

You may have noticed references to *RWRITE above. This command controls the writing of data to sideways RAM banks and the sideways RAM write-protection system. It is described in detail (along with all of ARIES-B32's commands) in the Technical Reference Guide section of this manual. More examples of its use are given in the ARIES-B32 Applications section of this manual.

Extended Buffers

To start, ensure that your system is configured with 20K of shadow RAM and 12K of unused sideways RAM in bank 14.

Create an extended printer buffer using *BUFFER:

```
 *BUFFER PRINTER 14{RETURN}
```

Use *XSTATUS S to check that the buffer is installed correctly. You should see something like:

```
 9: ROM 8K DFS 0.90
12: ROM 16K BASIC
14: RAM 12K Buffer PRINTER
15: ROM 16K ARIES-B32 1.00
RWRITE OFF
```

Put some characters into the buffer by typing {CTRL}{B} then typing a few lines of rubbish (ignore BASIC's error messages!), finishing off with {CTRI}{C}. Use *XSTATUS B to check the status of the extended buffer. You should see something like:

```
Buffer: PRINTER
Capacity: 11788 bytes
Contents: 177 bytes
PURGE OFF
```

Normal buffers are cleared whenever you press {ESCAPE} (unless you have specifically disabled this effect using *FX 230). Try pressing {ESCAPE} , then take another look at the extended buffer's status. You will see that it has not been cleared.

Since it may take a very long time to empty an extended buffer, it is undesirable for it to be as easy to clear as it is for a normal buffer. ARIES-B32's extended buffer system provides a means to control the emptying of the extended buffer independently of the normal buffers.

The last line of the *XSTATUS display holds the key to this. Currently it reads:

```
PURGE OFF
```

This means that {ESCAPE} will not cause the buffer to be purged (emptied). You can purge the extended buffer by typing:

```
*PURGE{RETURN}
```

*XSTATUS shows that the extended buffer is now empty. Put some more characters into it. Now type:

```
*PURGE ON{RETURN}
```

The last line of the *XSTATUS display shows that the command has been acted upon. Press {ESCAPE}. *XSTATUS will show that the buffer has been emptied.

Depending on which type of buffer you extend, the extended buffer will start up with purging on or off. A full list of which buffer starts up in which state is given in the reference section of this manual.

Press {CTRL}{BREAK}. You will find that, unlike sideways ROM images, the extended buffer is removed.

Type:

```
  *BUFFER PRINTER 14 UPPER{RETURN}
```

*XSTATUS S shows:

```
   9: ROM 8K DFS 0.90
  12: ROM 16K BASIC
  14: RAM 8K Unused
          4K Buffer PRINTER
  15: ROM 16K ARIES-B32 1.00
  RWRITE OFF
```

This version of the *BUFFER command gives you the ability to load an 8K ROM image in a sideways RAM bank and still use the upper half of the bank as an extended buffer. Note that the ROM image and the buffer may be loaded in either order.

Utilities

Ensure that your system is configured with 20K of shadow RAM by using *XON. Enter the following BASIC program:

```
    10 REM Dual screen system in MODE 4
    20 MODE 4
    30 MOVE 0, 511
    40 FOR I% = 1 TO 31
    50 DRAW I%*40, 1023
    60 DRAW 1279, 511
    70 DRAW I%*40, 0
    80 DRAW 0, 511
    90 NEXT I%
    100 PRINT TAB(13,15); "Now you see it!";
    110 *XMOVE V.5800 8000 V.3000
    120 CLS
    130 PRINT TAB(13,15); "Now you don't!";
    140 REPEAT
    150 TIME = 0 : REPEAT : UNTIL TIME > 99
    160 *XSWAP V.5800 8000 V.3000
    170 UNTIL FALSE
```

Save the program, then run it. Lines 20 to 100 set up a graphics screen display. Line 110 contains a special ARIES-B32 utility command: '*XMOVE'. We have already seen this command above, when we used it to save some copies of our tiny ROM.

*XMOVE copies the contents of one area of RAM into another. A full description of this command is given in the Technical Reference Guide, later in this manual. In line 110, *XMOVE is being used to copy the part of RAM displayed on the MODE 4 screen into the 'spare' 10K of RAM not used for the screen in MODE 4.

Lines 120 and 130 create a (rather simple) second screen display, replacing the original contents of the RAM displayed on the screen.

Lines 140 to 170 are a loop which repeats indefinitely with a one-second delay each time round. Line 160 contains a command which we have not seen before: '*XSWAP'.

*XSWAP is very similar to *XMOVE, except that it swaps the contents of one area of RAM with those of another. In line 160, *XSWAP is being used to swap the first screen display with the second.

To see another use of *XMOVE, type:

```
   *XOFF{RETURN}
```

Press {BREAK). Try and recover your program using OLD. As we saw above, BASIC cannot find the start of the program. We can recover from this situation using *XMOVE. Type:

```
  PRINT ~PAGE+&100, -PAGE{RETURN}
```

(PAGE is a special BASIC variable which contains the address of the base of program memory. &100 is the length of ARIES-B32's workspace and, therefore, the amount by which the base of program memory has moved due to *XOFF.)

Two hexadecimal numbers will be displayed, something like: 1A00
        1900

The precise values will depend on your particular system. If they are different from our example, substitute the values you got into the next command:

```
  *XMOVE P.1A00 7C00 P.1900{RETURN}
```

Now try OLD again. You will discover that your program has been moved so that BASIC can find it. A similar trick may be used to shift a program up before using *XON:

```
  *XMOVE P.1900 7B00 P.1A00{RETURN}
  *XON{RETURN}
  {BREAK}
  OLD{RETURN}
```

*XMOVE and *XSWAP have very many uses in a system with as many areas of RAM as a BBC micro fitted with ARIES-B32 and ARIES-B12. Further examples are given in the section on ARIES-B32 Applications later in this manual.

Technical Reference Guide

Command Summary

There are several new operating system commands introduced by ARIES-B32. Like most such commands and BASIC keywords, ARIES-B32 commands and their parameters may be abbreviated using a dot (for instance, ' *BUF. PR. 0' for '*BUFFER PRINTER 0'). Because of the high risk of clashes with other sideways ROM software, this manual does not show the use of abbreviated commands and we recommend that you avoid abbreviating ARIES-B32 commands in your programs. It is, however, quite reasonable to use abbreviations when you are typing commands in directly.

***XOFF**
***XON**
Control allocation of RAM in system. Options allow shadow RAM to be switched in and out without pressing {BREAK) or moving PAGE.

X
Used when shadow RAM is active to execute commands that need direct access to screen RAM.

***BUFFER**
Allows any buffer (printer, keyboard, serial in, serial out, sound, speech) to be extended using sideways RAM.

***PURGE**
Purges an extended buffer OR determines whether an extended buffer is purged when {ESCAPE) is pressed.

***RLOAD**
Loads a file into a sideways RAM bank.

***RWRITE**
Write-enables a bank of sideways RAM on ARIES-B32/ARIES-B12, or external sideways RAM. Write-protects all other sideways **RAM.**

***RKILL**
Disables ROMS on {BREAK). Removes ROM images from sideways RAM. ROMs return to their normal state next time {BREAK) is pressed.

***XSTATUS**
Gives reports of the system memory status, including: MODE, PAGE and HIMEM; full details of sideways ROMs and RAM; information on extended buffers.

***XMOVE**
***XSWAP**
Move or swap the contents of specified areas of the memory, including sideways ROM/RAM, shadow RAM and screen RAM.

***OSCLI**
Minimal language which passes commands directly to the operating system.

***XTEST**
Full function test of ARIES-B32 and ARIES-B12.

***HELP ARIES**

You can see a reminder of the ARIES-B32 commands at any time by typing:

```
  *HELP ARIES{RETURN}
```

Shadow RAM

Because ARIES-B32 permits you to use the same RAM for a variety of purposes, you must always remember that you are trading off shadow RAM against sideways RAM. The more you use of one, the less is available of the other.

ARIES-B32 is designed to be completely foolproof. If you try to use RAM that is not available, you will be warned with an error message. Remember: use *XSTATUS to determine what is wrong.

```
*XON (opt (page))
```

This command turns on shadow RAM in a specified configuration.

'opt' may take the values '20' or '16', indicating the amount of shadow RAM required in Kbytes. If 'opt' is not given, ARIES-B32 will take 20K of shadow RAM, except when the top 4K of sideways RAM bank 14 is in use, in which case only 16K of shadow RAM will be taken. If any of sideways RAM bank 13 is already in use for another purpose, the command will fail.

'page' is a hexadecimal number indicating the address of the page of RAM to be used for ARIES-B32 workspace (this is known as 'static' workspace and is, of course, illegal and unsafe). If a value of 0 is given, then normal sideways ROM workspace is allocated by the operating system (this is known as 'dynamic' workspace). If 'page' is not specified, and workspace is already present, then that workspace is used, otherwise dynamic workspace is created. If a nonsensical value of 'page' is given, the command will fail.

If your use of *XON causes a considerable re-configuration of your computer, the ARIES-B32 software will force you to press {BREAK) to give the operating system the opportunity to adjust to the new setup. If this is the case, the message:

```
  B32: Press BREAK to continue:
```

will appear, and your BBC Micro will wait until you press {BREAK).

{BREAK} will be demanded if, and only if:

1) Workspace is currently in use and the command asks for workspace elsewhere, or

2) Dynamic workspace is in existence and the command asks for static workspace, or

3) There is no workspace in existence and the command asks for dynamic workspace (possibly by default).

Conversely, {BREAK} will not be demanded if, and only if:

1) Workspace is currently in use and the command asks for workspace of the same type in the same place (possibly by default), or

2) There is no workspace in existence and the command asks for static workspace.

If the command does not demand {BREAK), it will take effect on the next screen MODE instruction (VDU code 22) or {BREAK). If you change your mind and enter another *XON command, or *XOFF, this will supersede the first one, with the most recently-specified value of 'page' being used as a default.

Examples (starting with shadow RAM off and no workspace):

```
  *XON          20K shadow RAM, dynamic workspace
  *XON 16       16K shadow RAM, dynamic workspace
  *XON 20 C     20K shadow RAM, workspace at &C00
```

***XOFF (opt)**

This command turns off shadow RAM.

'opt' may take the values '0', '1' or '2', specifying the detailed action of the command. If 'opt' is not given, a value of '0' is assumed. The action of *XOFF is as follows:

Opt	Action
0	Switch off shadow RAM totally and throw away workspace. You will be forced to press {BREAK).
1	Disable shadow RAM on the next screen MODE instruction (VDU code 22) or {BREAK}. Once the shadow RAM is disabled, the workspace is retained, but it may be overwritten without crashing the system. This option is useful for applications such as games.
2	Disable shadow RAM on the next screen MODE instruction (VDU code 22) or {BREAK). Once the shadow RAM is disabled, the workspace is retained, but it must not be overwritten until the next {BREAK}. This is the 'clean' way to switch shadow RAM off, as it does not disturb the rest of the system.

***X* command line**

The command line is executed as a normal '*-command', except that screen RAM is switched in in place of shadow RAM while the command is being executed.

Examples:

```
  *X*SAVE PICTURE 3000 8000     *SAVE a screen image
  *X*PDUMP                     Dump screen to printer using a
                               printer driver ROM
  *X*MOVEPOINTER               Move pointer in AMX Mouse system
```

NOTE: Do not use this command to enter a language. You should always use *XOFF if you wish to disable shadow RAM for an extended period.

Start-up Link

Close to the notched or dotted end of the processor chip on ARIES-B32 is a small link joining two pins (see figure 2). If you wish ARIES-B32 to start up with shadow RAM active when you turn your BBC Micro on or press {CTRL} {BREAK}, then you should leave this link in place. If you wish ARIES-B32 to start up with shadow RAM inactive, remove the link (you can store it on one of the two pins for safe keeping).

Note that, in the case of {CTRL}{BREAK} with the start-up link fitted and ARIES-B32 RAM in use for sideways ROM images, ARIES-B32 will only switch on as much shadow RAM as it can take without corrupting the sideways RAM.

Sideways ROM/RAM

***RLOAD filename bank**

Loads a file containing a sideways ROM image into a bank of sideways RAM.

'bank' is a decimal number which defines which sideways bank is to be used for the ROM image. The command will fail if:

- there does not appear to be any RAM in the specified area of the specified bank.
  the RAM in the specified bank is in use for some other purpose.
- the ROM image is too long for the available RAM.

If the RAM in the specified bank is in use as a sideways ROM image, the previous image will be overwritten, and you will be forced to press {BREAK) so that the operating system can adjust to the loss of the previous ROM.

If *RLOAD succeeds in loading a ROM image, it prints a message to remind you that the new ROM image will not be recognised by the operating system until you press {BREAK).

Some ROMs, particularly language ROMs, may be installed by a 'back-door' method, avoiding the need to press {BREAK} at this point. You will have to experiment to see whether this works for any individual ROM. The command you must type varies according to which sideways RAM bank you have used for the ROM image.

For sideways RAM bank 14, after loading the ROM image, type:

```
*XMOVE 514.8006 +1 P.2AF(RETURN}
```

For sideways RAM bank 13, after loading the ROM image, type:

```
*XMOVE 513.8006 +1 P.2AE{RETURN}
```

For sideways RAM bank 0, after loading the ROM image, type:

```
*XMOVE 50.8006 +1 P.2A1{RETURN}
```

***RWRITE bank**

By default, all sideways RAM on a system fitted with ARIES-B32 is write-protected. *RWRITE controls this write-protection.

'bank' controls the action of *RWRITE as follows:

Bank	Action
Decimal number	Causes data written to the sideways ROM/RAM area to be directed to the specified bank of sideways RAM on ARIES-B32 or ARIES-B12, regardless of which sideways ROM is active (this is known as ' write-through').
'EXTERNAL'	Write-enable any sideways RAM external to ARIES-B32.
'OFF'	Write-protect all sideways RAM.

The *RWRITE state is reset to 'OFF' by {CTRI}{BREAK}, *XON and *XOFF, otherwise it is preserved.

***RKILL romspecifier (romspecifier ...)**

Disables the specified ROM images.

'romspecifier' is either an actual sideways ROM bank number, or the name of a ROM as listed by *XSTATUS (not including the version number) . ROM names may be abbreviated using a dot. If there is more than one ROM whose name matches 'romspecifier', the matching ROM in the highest-numbered socket will be disabled. A list of ROM specifiers may be given, separated by spaces. If a ROM specifier itself contains spaces, it may be enclosed in double quotes.

*RKILL forces you to press {BREAK) so that the operating system has the opportunity to adjust to the removal of the ROMs.

If the sideways ROM image is in sideways RAM, it will be removed completely, otherwise the ROM will be re-activated when you next press {BREAK).

Examples:

```
  *RKILL BASIC VIEW        Disable BASIC and VIEW
```

Extended Buffers

***BUFFER OFF | name bank (position)**

Controls the use of sideways RAM as an extended buffer.

'*BUFFER OFF' removes the extended buffer. If you attempt to change the state of a buffer which is not empty, the command will fail.

'name' specifies which buffer is to be extended. The full list of options is:

KEYBOARD	
SERIN	(RS423 input)
SEROUT	(RS423 output)
PRINTER	
SOUND	(Four equal buffers for SOUND channels 0 - 3)
SPEECH	

'bank' is a decimal number which defines which sideways bank is to be used for the buffer. If there is no RAM in the specified area of the specified bank, the command will fail. If the RAM in the specified area of the specified bank is in use for some other purpose, the command will fail.

'position' is an optional parameter which allows a single bank of sideways RAM to be used for more than one purpose at once. The only permitted value is 'UPPER', which specifies that the buffer should occupy the upper half of the sideways RAM bank, leaving the lower half free for other uses.

NOTE: The extended buffer software lives in the RAM previously used for the buffer, so you must not overwrite the normal buffer area.

Examples:

```
  *BUFFER PRINTER 0
  *BUFFER SOUND 14 UPPER
```

***PURGE (opt)**

Controls clearing of the extended buffer on {ESCAPE}.

'opt' may take the values 'OFF' and 'ON'. If 'OFF' is specified, the extended buffer will not be purged when {ESCAPE) is pressed. If 'ON' is specified, the extended buffer will be purged when {ESCAPE} is pressed. If 'opt' is omitted, the extended buffer will be purged immediately, and its reaction to {ESCAPE} will be left unchanged.

The different buffers start up in different purge states when extended, as follows:

KEYBOARD	ON
SERIN	OFF
SEROUT	OFF
PRINTER	OFF
SOUND	ON (Cannot be turned off)
SPEECH	ON

Note that *FX 230 may be used to disable all of the {ESCAPE} actions, including buffer purging.

Utilities

*****OSCLI**

The ARIES-B32 ROM contains a minimal language, 'OSCLI', which takes the commands you type and passes them straight to the operating system. This language is entered automatically if you *RKILL the current language, or start your computer up with no other language fitted. It may be entered explicitly with the command *OSCLI. The prompt for OSCLI is

In addition to its use as a security measure for *RKILL, OSCLI permits you to have a system with no 'real' language fitted, and load any language you require from disc, using *RLOAD.

***XMOVE source start source end destination start**
***XSWAP source start source end destination start**

Moves/swaps data between areas of memory in the BBC Micro's standard RAM, sideways ROM/RAM banks and shadow RAM.

'source start' 'source end' and 'destination start' are hexadecimal addresses, using a prefix separated from the address by a dot to indicate which bank of memory is required where a choice exists. The data area starts at 'source start' and finishes with the byte before ' source_end' (in the same way as *SAVE). You should not use areas which cross the &8000 boundary. The address prefixes are:

P:	Program/data RAM (this will be the same as video RAM if shadow RAM is off)
V:	Video RAM
Sn:	Sideways ROM/RAM ('n' is decimal bank number)

The hexadecimal length of the data area may be specified instead of the end address by using '+' as in DFS. For example:

```
*XMOVE P.2000 +2000 V.3000
```

is identical to:

```
*XMOVE P.2000 4000 V.3000
```

Examples:

```
*XMOVE 514.8000 C000 P.3000        Copy 16K sideways RAM down
*XMOVE 53.8000 +2000 50.8000       Copy 8K ROM into sideways RAM
*XMOVE P.1900 7F00 P.1A00          Move program up one page
*XSWAP P.3000 8000 V.3000          Swap graphics screens
```

***XSTATUS (opt ...)**

Gives report of system status.

'opt' specifies which aspects are to be displayed. More than one aspect may be specified, by giving several options separated by spaces.

Opt	Action
X	Reports on shadow RAM configuration, present and pending
S	Reports on sideways ROM/RAM system: gives ROM titles, version numbers and sizes present; indicates inactive ROMs (such as those de-activated by *RKILL) by showing their names in brackets; says which banks are RAM and which ROM; shows usage of RAM banks or part-banks which do not contain ROM images
B	Reports on extended buffer system

***XTEST (opt)**

Performs a thorough test of ARIES-B32 and ARIES-B12 (if it is present) and prints the results.

'opt' is an optional parameter which specifies a special action. The only permitted value is 'REPEAT', which causes the test to be repeated indefinitely (or until it fails).

Machine Code Interface

ARIES-B32 is designed to be completely transparent to software that uses the correct MOS interfaces. In general you may modify the various operating system vectors freely as long as you use the correct methods of doing so (see the Advanced User Guide for details of this). The one exception is that interrupt routines (including event routines) must be located below &3000 when shadow RAM is in use.

When ARIES-B32 shadow RAM is deactivated by *XOFF 0, it resets OSHWM to the value it would have if ARIES-B32 was not fitted to the computer. When ARIES-B32 shadow RAM is active, or deactivated by *XOFF 1 or *XOFF 2, it uses 256 bytes of private workspace and thus increases OSHWM by one page.

When shadow RAM is active, the ARIES-B32 shadow RAM switch state may be explicitly controlled by an OSBYTE call which has been officially assigned for this purpose. This enables programs to read and write data in either bank of RAM. The ARIES shadow RAM OSBYTE call may also be used to test for the presence of active shadow RAM.

Block copying and swapping of memory contents may be carried out within machine code programs by use of the ARIES-B32 XMOVE/XSWAP OSWORD call.

Other aspects of ARIES-B32 configuration may be controlled by use of the normal ARIES-B32 commands via OSCLI. If a program containing ARIES-B32 specific commands is required to run without ARIES-B32 fitted, you will need to intercept the BRK vector to handle the unrecognised command errors, or, preferably, use the ARIES shadow RAM OSBYTE to check for ARIES-B32's presence before attempting to issue such commands.

OSBYTE with A=&6F   Read/Write Shadow RAM Switch

On entry:

```
   A  &6F
   X  determines action as follows:

   Bit 6 - 0       write switch state
           1       read switch state

   Bit 7 - 0       no stack operation
           1       read operations: pop state from stack
                   write operations: push state onto stack

   Bit 0 - 0       select video RAM
           1       select shadow RAM

   All other bits must be set to 0.
```

On exit, X indicates the previous switch state:

```
   Bit 0 - 0       video RAM
           1       shadow RAM
```

The ARIES-B32 software maintains a record of past switch states on its own 8-deep stack. This allows a program to write the state in order to perform some operation, and then restore the previous state automatically with a read operation. For example:

```
   LDA #&6F
   LDX #&81             Select shadow RAM, saving previous
   JSR OSBYTE           state on stack
     .
     .                  Perform some operation
     .
   LDA #&6F
   LDX #&C0             Read current state, restoring previous
   JSR OSBYTE           state from stack
```

In particular, this allows convenient access to either area of RAM from within interrupt routines.

OSWORD with A=&45   Move/Swap Memory Contents

This routine takes a 10-byte parameter block, addressed by the X and Y registers. For more details and examples of its use, see *XMOVE and *XSWAP above.

The parameter block is constructed as follows:

```
XY+   0 - Operation type: 0 = Move, 1 = Swap
      1 - Source start address, low byte
      2 - Source start address, high byte
      3 - Source bank specifier
      4 - Source end address + 1, low byte
      5 - Source end address + 1, high byte
      6 - Destination start address, low byte
      7 - Destination start address, high byte
      8 - Destination bank specifier
      9 - Return code
```

The bank specifiers are constructed as follows:

```
Bits: 76543210
          ****  - Sideways bank number
       ***      - Must be set to 0
      *         - 0 = program RAM, 1 = video RAM
```

Following a call of this routine, the return code field of the parameter block will indicate the results of the call as follows:

Value	Meaning
0	Successful call
1 - 4	Error - call failed
1	Overlapping areas for swap
2	Source end before source start or source end > &FFFF
3	Destination end > &FFFF
4	Attempted write to non-existent RAM in bank 13 or 14

Effects on the MOS Interface

The following OSBYTE call must be used to determine the start of available program/data RAM (this is essential for compatibility with any sideways ROM software, not just ARIES-B32):

OSBYTE with A=&83	Read top of OS RAM address (OSHWM)

The values returned by the following OSBYTE calls depend on the current and pending shadow RAM states:

OSBYTE with A=&84	Read current top of language RAM
OSBYTE with A=&85	Read top of I/O processor available RAM for specified mode

Note that the value returned by the following OSBYTE call is a 2-byte number with the low byte in X and the high byte in Y:

OSBYTE with A=&80	Get buffer status

In addition, the following MOS calls are reserved for use by ARIES-B32 (these definitions are officially approved by Acorn):

OSBYTE with A=&6F	Read/Write Shadow RAM Switch (see above)
OSBYTE with A=&EF	Read/write location &27F (this call is used at a low level by the BBC micro model B+ to control shadow RAM)
OSWORD with A=&45	Move/Swap Memory Contents (see above)

Memory Locations Used by ARIES—B32

When shadow RAM is active or temporarily disabled by *XOFF 1 or *XOFF 2, one page of RAM is used by ARIES-B32 for workspace to control switching of shadow RAM. This page is normally in the sideways ROM private workspace area, but it may be moved to any sensible location by the user (see *XON above).

ARIES-B32 uses the OSvariable at location &27F for internal status information (this location is used for a similar purpose by the BBC micro model B+).

Locations from &100 upwards are used as (very) temporary workspace when processing some ARIES-B32 commands and machine-code calls.

Many of the MOS vectors may be intercepted by ARIES-B32, depending on configuration (this does not include those which are unused in MOS 1. 2). All of these vectors are passed on in the correct fashion.

When an extended buffer is active, the original buffer is used as workspace for the extended buffer system.

Testing for ARIES—B32's Presence

It is a fairly straightforward matter to test for the presence of ARIES-B32 from BASIC and other languages with the equivalent of the ON ERROR facility. The only disadvantage is that systems which have a disc filing system will perform a disc access, which means that they will need to have a disc present in the current drive when the test is performed.

The following program fragment illustrates the technique:

```
10 aries = TRUE
20 ON ERROR aries = FALSE : GOTO 40
30 *XMOVE P.2000 +1 P.2000
40 ON ERROR OFF
50 IF aries THEN PRINT "Present" ELSE PRINT "Absent"
```

Line 20 sets up an error handler which is executed if *XMOVE is not recognised in line 30. If no error is generated, then the command has been recognised and we may infer that ARIES-B32 is present.

Testing for Active Shadow Ram

You can test whether shadow RAM is active quite simply, by calling the ARIES-B32 OSBYTE function with X set to &40 (this avoids altering the state in any way) and checking the value of the processor overflow flag ('V') on return. If shadow RAM is not active, the OSBYTE call will not be recognised and the overflow flag will be returned set.

To perform the call from BASIC, use the USR function, which will return a 4-byte number whose top byte is the contants of the Processor Status Register, P, on exit from the call.

Sideways RAM Use Markers

In order that ARIES-B32 may regulate the use of sideways RAM and protect you from accidentally overwriting important data, a system of Use Markers is provided.

Sideways RAM is sub-divided into lower and upper 8K sections, starting at &8000 and &A000, respectively. A pointer is stored, offset 7 bytes from the base of each section (i.e. at &8007 and &A007), which gives the offset from the base of the section to a 4-byte Use Marker. Valid Use Markers are as follows:

Hexadecimal	ASCII	Meaning
&65657246	'Free'	Unused (free for use)
&29432800	' (C)'	In use as sideways ROM image
&66667542	'Buff'	In use as extended buffer
&72657355	'User'	In use as user program/data

If there is not a valid Use Marker in a section starting at &8000, ARIES-B32 will assume that the section is not in use and may be overwritten. If there is not a valid Use Marker in a section starting at &A000, ARIES-B32 will assume that the section is in use for the same purpose as the section starting at &8000 in the same sideways RAM bank (i.e. that the whole 16K bank is used for the one purpose).

If you wish to make use of a section of sideways RAM for some purpose of your own, it is your responsibility to ensure that it is available for use by checking for the 'Free' Use Marker. You should mark any sections of sideways RAM which you are using in this way with the ' User' Use Marker in the following form:

Offset	Contents
0 - 3	Use Marker (as above) 'User' 4 Length of the area in use in Kbytes (&8004 may only take the values 8, 12 or 16; &A004 may only take the values 4 or 8)
5 - 6	Reserved - must be set to 0
7	Offset from base to Use Marker, in this case 0

NOTE: the Use Markers written by ARIES-B32 may not be at the same addresses as the above example; you should always use the pointer at offset 7 to find the Use Marker. ARIES-B32 does not provide the length information with its Use Markers; it is vital that you do so to allow for future software developments.

If you are using the whole of a bank of sideways RAM, you need not place a second Use Marker at &A000, but you should ensure that the previous Use Marker there is overwritten.

When you have finished with the section of sideways RAM, you must mark it as free. If you were using the whole of the bank, you must restore the Use Marker of the upper section.

Testing for Free Sideways RAM

The following program fragment shows one method of extracting and testing the sideways RAM Use Marker:

```
     10 DIM buff 255
     20 OSCLI ("*XMOVE S14.8000 8100 P." + STR$~(buff))
     30 inuse% = (buff!(buff?7) <> &65657246)
     40 IF inuse% THEN PRINT "RAM in use"; : STOP
```

Line 10 defines a buffer in program/data RAM. Line 20 copies the first 256 bytes of sideways RAM bank 14 into the buffer. Line 30 tests the value of the Use Marker and sets up a flag accordingly. Line 40 prints an error message and halts the program if the RAM is in use.

Marking Sideways RAM as In Use

Continuing the previous example, this program fragment goes on to mark the first 8K of sideways RAM bank 14 as used:

```
     50 *RWRITE 14
     60 $&8000 = "User"
     70 !&8004 = 8
     80*RWRITE OFF
```

Line 50 sets up write-through to sideways RAM bank 14. Lines 60 and 70 write in the Use Marker. Line 80 switches off write-through to sideways RAM.

The final program fragment illustrates how you can mark the sideways RAM as free when you have finished with it:

```
   500 *RWRITE 14
   510 $&8000 = "Free"
   520 !&8004 = 8
   530 *RWRITE OFF
```

Further examples of the use of sideways RAM Use Markers are to be found in the section of this manual on 'ARIES-B32 Applications'.

Error Messages

Error messages produced by ARIES-B32 are prefixed with 'B32: with the following exceptions: the escape message is printed in the standard form; *RKILL may cause the MOS error message 'Bad string'; *RLOAD may cause various filing system errors.

All of the ARIES-B32 error messages are listed below, in alphabetical order, together with their error numbers. The error numbers are not normally displayed when an error is reported, but you may use them to write your own error handling programs.

132 Bad bank

An attempt was made to use a sideways ROM/RAM bank which does not exist, or an attempt has been made to use a sideways bank which is not recognised as RAM by ARIES-B32.

145 Bad buffer

An attempt was made to refer to a buffer which does not exist.

128 Bad option

One of the parameters of the command given was not recognised.

129 Bad page

An incorrect static workspace address was given with *XON.

140 Bad region

The region of memory specified in *XMOVE or *XSWAP had a negative length or exceeded the address space of the processor.

130 Bad syntax

The parameters of the command given did not make sense.

136 Buffer in use

An attempt was made to extend or return to normal a buffer which was not empty.

137 Extended buffer already present

An attempt was made to extend a buffer when an extended buffer was already in existence.

214 File not found

The file referred to was not found.

139 Line too long

The command given was too long. This error may be avoided by placing the command outside shadow memory.

143 No extended buffer present

An attempt was made to refer to an extended buffer when no extended buffer existed.

144 Not allowed with SOUND

*PURGE OFF is not permitted with an extended SOUND buffer.

133 Not enough RAM

An attempt was made to *RLOAD a file which was too long for the sideways RAM available.

135 Not RAM

An attempt was made to write to a sideways bank which ARIES-B32 does not recognise as RAM.

138 Not ROM

An attempt was made to *XKLUDGE a sideways bank which did not contain a valid sideways ROM.

141 Overlapping areas

An attempt was made to *XSWAP two areas of memory which overlap.

131 RAM in use

An attempt was made to use ARIES-B32 RAM or sideways RAM which was already in use.

134 ROM in use

An attempt was made to overwrite the current filing system ROM by *RLOAD.

142 ROM not found

The ROM referred to could not be found.

Error Codes

128	Bad option
129	Bad page
130	Bad syntax
131	RAM in use
132	Bad bank
133	Not enough RAM
134	ROM in use
135	Not RAM
136	Buffer in use
137	Extended buffer already present
138	Not ROM
139	Line too long
140	Bad region
141	Overlapping areas
142	ROM not found
143	No extended buffer present
144	Not allowed with SOUND
145	Bad buffer
214	File not found

Using ARIES—B32 with Specific Software

General

In normal circumstances, either a piece of software will work with shadow RAM active, or it will not. All you have to do is use *XON and *XOFF to set your system up before running any particular program.

Some pieces of software have the unusual characteristic of being, in principle, capable of using shadow RAM, but, in fact, failing to work satisfactorily when shadow RAM is active. Other programs may interact with ARI ES-B3 2 ' s control software in undesirable ways. This section of the manual describes some simple solutions you can apply to problems with specific pieces of software.

It maybe that the piece of software you are interested in is not amongst those discussed below. In that event, you may still find that the techniques suggested will work equally well in your particular case.

Acorn ADFS

Acorn's Advanced Disc Filing System (ADFS) uses its workspace in a rather unusual way. ADFS attempts to keep any open files open over {BREAK}. In order to do this it requires that its workspace should not move over {BREAK}.

When you use *XOFF or *XON in such a way that ARIES-B32 gives up or takes workspace you are forced to press {BREAK}. If ARIESB32's ROM is in a higher priority socket than the ADFS ROM, this will cause ADFS's workspace to move, resulting in a 'bad sum' error message from ADFS and the computer hanging up until you press {CTRL}{BREAK}.

The solution is quite simple. Move the ROMs in your system so that ADFS is in a higher-numbered socket than ARIES-B32's ROM. Remember that you may put any ROM into the socket on ARIES-B32. It will usually be simplest to swap ADFS with ARIES-B32's ROM.

Acornsoft View A2.1

View A2.1 has a well-publicised bug, which causes it to fail to scroll the screen correctly when using shadow RAM.

One solution is to upgrade your View A2.1 to the latest version, View 3.0, which features a number of improvements over View A2.1. As an alternative, ARIES-B32 provides a special command which copies the ROM image into sideways RAM and patches it to correct the bug.

To patch View A2.1, you will need to have a bank of 16K of sideways RAM available. This can be achieved by using *XON 16 to split ARIES-B32's RAM into 16K shadow RAM and 16K sideways RAM in bank 14.

The special command '*VKLUDGE' finds View A2.1 automatically, regardless of which sideways socket it is fitted into, and places the patched version into the sideways RAM bank you specify.

A complete sequence of commands to place a patched copy of View A2.1 in sideways RAM bank 14 is:

```
  *XON 16{RETURN}
  MODE 3{RETURN}
  *VKLUDGE 14{RETURN}
  {BREAK}
```

If you are using a bank of sideways RAM whose number is higher than that of the original ROM, then you can enter the modified copy of View directly, in the normal way. If, however, you are using sideways RAM in bank 0 for the patched copy, you should first use *RKILL to disable the original ROM, as otherwise the operating system will take the higher-numbered socket, containing the incorrect version.

Computer Concepts ROMs

Most of the current Computer Concepts ROMs are capable of working with shadow RAM systems. If you find that you have a version of a Computer Concepts ROM which does not appear to function correctly with shadow RAM, you should contact Computer Concepts for an upgrade to the latest version.

Wordwise-Plus

Note that you will need Wordwise-Plus version 1.4D or later for the technique described here to work.

Because it was written before shadow RAM systems were developed, Wordwise-Plus is a special case. There is no way of using shadow RAM to allow you to edit text in an 80-column screen mode (you will need INTER-WORD for that!). However, it is possible to use ARIES-B32 shadow RAM to preview long documents in 80-column format.

You will need to program two function keys as follows:

```
  *KEY 0 *XOFF 2|M|M|[{RETURN}
  *KEY 7 *XON|M|M7 {RETURN}
```

To go from the main menu to editing your document, press {f0} (function key 0). To return to the menu after editing your document, press {ESCAPE} in the normal way. To preview your document in 80-column format, press {f7}.

ROMs Written Specifically for ARIES—B20

Very much against our advice, some ROMs have been written which rely on the shadow RAM hardware being identical to ARIES-B20. These ROMs will not work with ARIES-B32 without modification.

ARIES-B32 provides a special command which copies a ROM image into sideways RAM and alters any parts of the software which explicitly refer to ARIES-B20 hardware to work correctly with ARIES-B32 hardware.

Depending on the size of the ROM you wish to patch, you will need to have a bank of either 8K or 16K of sideways RAM available. If necessary, you can use *XON 16 to split ARIES-B32's RAM into 16K shadow RAM and 16K sideways RAM in bank 14.

The command '*XKLUDGE' finds the ROM whose name you specify (in the same way as in *RKILL), regardless of which sideways socket it is fitted into, and places the patched version into the sideways RAM bank you specify.

A complete sequence of commands to place a patched copy of Wordwise 1. 20 in sideways RAM bank 14 is:

```
  *XKLUDGE WORDWISE 14{RETURN}
  {BREAK}
```

If you are using a bank of sideways RAM whose number is higher than that of the original ROM, then you can enter the modified copy of the software directly, in the normal way. If, however, you are using sideways RAM in bank 0 for the patched copy, you should first use *RKILL to disable the original ROM, as otherwise the operating system will take the higher-numbered socket, containing the original, incorrect version.

ARIES—B32 Applications

The examples below are meant to be taken as outlines only, not as complete finished working programs. By using them as bases, you should be able to develop your own versions which do precisely what you want.

Saving the Contents of Sideways RAM

Although this is not an application of ARIES-B32 in itself, you will find that this technique is a useful adjunct to the examples below.

You must have shadow RAM active or be in screen mode 7 before attempting this, in order to avoid the risk of corrupting the data before it is saved to disc.

To save the bottom 8K of sideways RAM bank 14, type:

```
*XMOVE S14.8000 +2000 P.3000{RETURN}
*SAVE MYRAM 3000 +2000{RETURN}
```

To save the whole 16K of sideways RAM bank 0, type:

```
*XMOVE 50.8000 +4000 P.3000{RETURN}
*SAVE MYRAM 3000 +4000{RETURN}
```

Any programs or data between addresses &3000 and &7000 will be overwritten by these commands. Allowing for that, the commands may be included within programs.

If sideways RAM is to be saved when your program/data overlaps the area &3000 to &7000, without corrupting the program/data, you should use the following commands:

```
*XSWAP S14.8000 +2000 P.3000{RETURN}
*SAVE MYRAM 3000 +2000{RETURN}
*XSWAP S14.8000 +2000 P.3000{RETURN}
```

The second *XSWAP command restores your program/data.

If you wish to incorporate these commands within a program, you must ensure that the lines containing the commands are outside the area of RAM between &3000 and &7000.

Saving and Loading Very Long Programs

Some programs supplied on cassette are simply too long to fit into a machine with discs fitted, due to the fact that the disc filing system takes a certain amount of RAM for its own workspace. Other programs are written in machine code and will only run in a fixed area of RAM; often that area is occupied by the disc filing system workspace.

The example below shows how you can use *XMOVE and sideways RAM to allow you to load such programs from disc.

First, you need to save the program to disc. Enter the following BASIC program:

```
    10 REM Long program saver (C) 1985 ARIES Computers
    20 *XON 16
    30 MODE 7
    40 HIMEM = &4000
    50 *RWRITE 14
    60 *TAPE
    70 *LOAD MYPROG 4E00
    80 *DISC
    90 *SAVE PART1 4E00 8000
   100 *XMOVE 514.8000 BBF0 P.4000
   110 *SAVE PART2 4000 7BF0
   120 $&8000 = "Free"
   130 !&8004 = 8
   140 $&A000 = "Free"
   150 !&A004 = 8
   160 *RWRITE OFF
```

Lines 20 and 30 configure ARIES-B32 with 16K of shadow RAM and 16K of sideways RAM in bank 14.

Line 40 moves the top of BASIC's workspace down, leaving the top section of program/data RAM available as a buffer for the first part of the cassette program. Line 50 sets up write-through to sideways RAM bank 14, so it can be used as a buffer for the rest of the cassette program.

Lines 60 to 80 switch to the cassette filing system, load the program and then switch back to the disc filing system.

Line 90 saves the first 16K of the program to disc. Line 100 moves the rest of the program down into normal RAM from sideways RAM. Line 110 saves this copy of the rest of the program to disc.

Lines 120 to 150 mark sideways RAM bank 14 as unused (note that the program does not bother to mark the sideways RAM as in use, since the use is sufficiently temporary that there is no risk of this causing a problem). Line 160 switches off write-through to sideways RAM.

Save the program, then run it. You now have two files on disc which contain the two parts of your cassette program. To load them back in, you will need another program:

```
    10 REM Long program loader (C) 1985 ARIES Computers
    20 *XOFF 1
    30 MODE 7
    40 *LOAD PART2 4000
    50 *RWRITE 14
    60 *LOAD PART1 8E00
    70 *TAPE
    80 PAGE = &E00
    90 *KEY 0 *XMOVE S14.8E00 C000 P.E00|M$&A000="Free"|M!&A004=8
    IM*RWRITE OFFIM
   100 *FX 138,0,128
```

Lines 20 and 30 configure ARIES-B32 with shadow RAM switched off, so that its workspace may be overwritten by the cassette program.

Line 40 loads the second part of the cassette program from disc into the top part of program/data RAM.

Line 50 sets up write-through to sideways RAM bank 14, so it can be used as a buffer for the first part of the cassette program. Line 60 loads the first part of the cassette program from disc into sideways RAM bank 14.

Line 70 switches off the disc filing system, preparatory to our moving the first part of the cassette program so that it overwrites the filing system workspace. Line 80 informs BASIC that the base of program RAM is now &E00.

Because the first part of the cassette program is going to overwrite our BASIC program when it is moved into place, we must program the remaining commands into a function key, this is done in line 90. These commands copy the first part of the cassette program down into program/data RAM, then mark sideways RAM bank 14 as unused, then switch off write-through to sideways RAM.

Line 100 puts the code for function key 0 into the keyboard buffer, so that it will be executed when the program finishes.

Save this program, then run it. As it stands, it is not quite complete. If the cassette program was in BASIC, you should type:

```
  OLD{RETURN}
```

This will set the program up ready to be run. You can modify your loader program to do this, and run the program automatically. Use {CTRL}{BREAK} to return your system to normal. Load the loader program. Enter the following amended lines:

```
    10 REM Long BASIC program loader (C) 1985 ARIES Computers
    90 *KEY 0 *XMOVE S14.8E00 C000 P.E00|M$&A000="Free"|M!&A004=8
   |M*RWRITE OFF|MOLD|MRUN|M
```

Don't forget to save this version of the loader!

If the cassette program was in machine code, you will need to know what its execution address was (this can be found from the original cassette file). If, for instance, the program's execution address was &2000, you should make a version of the loader program with the following amended lines:

```
    10 REM Long machine code program loader (C) 1985 ARIES Computers
    90 *KEY 0 *XMOVE S14.8E00 C000 P.E00|M$&A000="Free"|M!&A004=8
  IM*RWRITE OFF|MCALL &2000|M
```

If you want to use shadow RAM with the cassette program, you will need to use *XON requesting static workspace (this is described in the Technical Reference Guide earlier in this manual). The *XON command may be built into the function key definition, but you will need to use *XOFF to dispose of ARIES-B32's workspace before running the program, to avoid being forced to press {BREAK} when you use *XON (again, this is described in detail in the Technical Reference Guide).

The above programs are examples only; they are not correct for all situations. You will have to work out how to adapt them to your precise requirements. Using variants of these basic techniques, however, will enable you to save and load a wide variety of 'difficult' programs.

Saving and Loading Programs Using Sideways RAM

While ARIES-B32 does not provide a RAM filing system as such, it may still be used as a high-speed program save and load area. If you are developing a program, it is often convenient to be able to save it temporarily while you experiment.

As an example, for a program less than 12K long, set up two function keys, one to save and one to load. First, you need to know where your program starts. Type:

```
  PRINT ~PAGE{RETURN}
```

A hexadecimal number will be displayed, which is the address of the start of your program. In disc-based systems with Acorn DFS, for example, the normal value with ARIES-B32 shadow RAM active would be 1A00.

Program the function keys according to the actual value printed:

```
  *KEY 0 *XMOVE P.1A00 +2FF8 S14.8008{RETURN}
  *KEY 1 *XMOVE S14.8008 +2FF8 P.1A00{RETURN}
```

(Note that the first 8 bytes of the sideways RAM are reserved for the Use Marker.)

Don't forget to mark the bank of sideways RAM as in use:

```
  *RWRITE 14{RETURN
  $&8000 = "User"{RETURN}
  !&8004 = 12{RETURN}
  ?&A007 = 7{RETURN}
  *RWRITE OFF{RETURN}
```

To save your program temporarily in sideways RAM, press {f0}, to recover the saved copy, press {f1}.

If you had two programs which you wished to use alternately, you could program a single key to swap them over:

```
  *KEY 2 *XSWAP P.1A00 +2FF8 S14.8008{RETURN}
```

You can also use this technique to produce program overlays in BASIC (this is a technique whereby programs longer than the program RAM may be run, by loading sections of them as they are needed). Overlaying like this is particularly useful when developing long assembler programs, such as sideways ROMs. Set up sideways RAM as above then enter the following program fragment:

```
  1000 DEF PROCoverlayl
  1010 PRINT "This is overlay one"
  1020 ENDPROC
```

Copy this overlay into sideways RAM:

```
*XMOVE P.1A00 +100 S14.8008{RETURN}
```

Make a second overlay:

```
   1000 DEF PROCoverlay2
   1010 PRINT "This is overlay two"
   1020 ENDPROC
```

Copy the second overlay into sideways RAM:

```
   *XMOVE P.1A00 +100 S14.A008{RETURN}
```

Create the main program:

```
     10 REM Overlay demonstration (C) 1986 Aries Computers
     20 LOMEM = &3000
     30 PAGE = &2F00
     40 *XMOVE 514.8008 +100 P.2F00
     50 PROCoverlayl
     60 *XSWAP S14.A008 +100 P.2F00
     70 PROC overlay2
     80 PAGE = &1A00
```

As an example of a practical use of this technique, you might make two 12K overlays and save each of them to disc (using *SAVE). You could load one overlay directly into the overlay area (in our example program this is 256 bytes long, from &2F00 to &3000: this would have to be adjusted!). The second overlay would be loaded directly into sideways RAM (using *RWRITE and *LOAD). The overlays could then be swapped using *XSWAP, as required.

Don't forget to mark sideways RAM that is in use for overlays. (The overlay example program should really have lines to do this added; you might like to try this as an exercise.)

When you have finished experimenting with sideways RAM, remember to mark it as unused:

```
  *RWRITE 14{RETURN}
  $&8000 = "Free"{RETURN}
  !&8004 = 8{RETURN}
  $&A000 = "Free"{RETURN}
  !&A004 = 8{RETURN}
  *RWRITE OFF{RETURN}
```

Swapping Between Screens

An example of the use of *XMOVE and *XSWAP to manipulate two screens is given in the Advanced Tutorial section of this manual. It is worth observing that shadow RAM or sideways RAM may also be used to store alternative screens.

Of course, you do not need to swap the entire screen. You might, for instance, swap in a small menu at the top of the screen, occupying just a few lines. This technique is particularly useful in terminal emulator programs.

You must remember that you are simply changing the contents of the display RAM when you swap screens using this technique. The screen mode, cursor position, etc. will not be changed.

There is one problem which arises from this limitation: scrolling. Due to the way the BBC micro works, when you scroll one screen, the other will also appear to have scrolled when you swap it into display RAM.

One method of avoiding this is to define a text window the same size as the screen. This forces the BBC micro to scroll the contents of RAM in a less efficient way, but one which does not cause the problem.

It is possible to control the video circuits directly, bypassing the operating system, and thus overcome the limitations of the simple approach. Such specialised programming is beyond the scope of this manual. If you require further information you should consult the Advanced User Guide.

High-speed Database Using Sideways RAM

If you are writing your own database program, you can use sideways RAM as a high-speed random-access storage area. The following example program shows the basic technique:

```
   10 REM High-speed database (C) 1986 ARIES Computers
   20 PROCinitialise
   30 REPEAT
   40 INPUT "Record number (0 - 100): " r%
   50 PRINT "       Old contents: "; FNget (r%)
   60 INPUT " New contents or RETURN: " s$
   70 IF (LEN (s$) > 0) THEN PROCput (r%, s$)
   80 UNTIL FALSE
   90 DEF PROCput (record%, contents$)
  100 $buff = contents$
  110 destination% = &8100 + (recordlength% * record%)
  120 putblock?6 = destination% MOD 256
  130 putblock?7 = destination% DIV 256
  140 PROCOSWORD (&45, putblock)
  150 ENDPROC
  160 DEF FNget (record%)
  170 source% = &8100 + (recordlength% * record%)
  180 sourceend% = source% + recordlength%
  190 getblock?1 = source% MOD 256
  200 getblock?2 = source% DIV 256
  210 getblock?4 = sourceend% MOD 256
  220 getblock?5 = sourceend% DIV 256
  230 PROCOSWORD (&45, getblock)
  240 =$buff
  250 DEF PROCOSWORD (number%, blockaddress%)
  260 A% = number% : X% = blockaddress% MOD 256 : Y% = blockadd
 ress% DIV 256
  270 CALL &FFF1
  280 ENDPROC
  290 DEF PROCinitialise
  300 recordlength% = 64
  310 DIM buff recordlength%
  320 DIM putblock 9
  330 putblock?0 = 0
  340 putblock!1 = buff
  350 putblock?3 = 0
  360 putblock!4 = buff + recordlength%
  370 putblock?8 = 14
  380 DIM getblock 9
  390 getblock?0 = 0
  400 getblock?3 = 14
  410 getblock!6 = buff
  420 getblock!8 = 0
  430 getblock?1 = &00
  440 getblock?2 = &80
  450 getblock?4 = recordlength%
  460 getblock?5 = &80
  470 PROCOSWORD (&45, getblock)
  480 IF buff!(buff?7) <> &65657246 THEN PRINT "RAM in use"; :
 STOP
  490 *RWRITE 14
  500 $&8000 = "User"
  510 !&8004 = 8
  520 *RWRITE OFF
  530 ON ERROR PROCtidy
  540 FOR I% = 0 TO 100
  550   PROCput (I%, "* empty *")
  560 NEXT I%
  570 ENDPROC
  580 DEF PROCtidy
  590 *RWRITE 14
  600 $&8000 = "Free"
  610 !&8004 = 8
  620 *RWRITE OFF
  630 REPORT
  640 STOP
  650 ENDPROC
```

Although it looks rather daunting, this program is quite easy to understand. It is well worth taking the time to work through this explanation of it, as it contains examples of many useful techniques.

Line 20 calls a procedure at line 290 which sets the system up. Line 300 defines the length of our database records. You can change this to suit your application. Line 310 sets up a buffer in program/data RAM which will hold a copy of a single record from the main database.

Lines 320 to 420 set up two parameter blocks for the *XMOVE OSWORD call. The program uses this OSWORD call rather than the *XMOVE command because it allows it to use variable parameters. A full description of this OSWORD call is given in the Technical Reference Guide section of this manual.

Lines 430 to 460 set up the parameter block which the program uses for retrieving data from sideways RAM to copy the sideways RAM Use Marker (again, this is explained in the Technical Reference Guide section) into the buffer. Line 470 calls a procedure which sets up parameters and calls OSWORD. Line 480 tests the sideways RAM Use Marker to check that the sideways RAM is free (note that the program uses only the bottom 8K of sideways RAM bank 14).

Lines 490 to 520 write a new sideways RAM Use Marker into sideways RAM bank 14, to indicate that the RAM is now in use. Line 530 sets up a procedure to be called whenever the program is exited due to an error or {ESCAPE}, which marks the sideways RAM as free.

Lines 540 to 560 write an initial value into all the records in the database. In this example, the records are strings. They might just as well be numbers, or a combination of several data in a single record.

Returning to the main program, lines 30 to 80 form a loop which repeats forever. In this loop, you are prompted to input a record number, the contents of the record you select are displayed and you are offered the opportunity to change the contents. You can exit from the loop at any point by pressing {ESCAPE}.

The program uses two procedures to put information into the database and retrieve it.

Lines 90 to 150 contain the definition of a procedure which writes a single record into the database. The procedure takes two arguments, giving the number of the record and the data to be written into that record. Line 100 copies the data into the buffer. Line 110 calculates the address of the record within the sideways RAM bank. Lines 120 to 140 set up and call the OSWORD call which actually copies the data from the buffer into sideways RAM.

Lines 160 to 240 contain the definition of a function which fetches a single record from the database. The function takes one argument, giving the number of the record to be fetched and returns the contents of that record. Line 170 calculates the address of the record within the sideways RAM bank. Line 180 calculates the address one byte after the end of the record. Lines 190 to 230 set up and call the OSWORD call which actually copies the data from the buffer into sideways RAM.

The program as presented here is very simple and rather restricted. You might add sorting and searching facilities to make it more useful. The *XMOVE OSWORD call permits very fast insertion sorts.

To increase the storage capacity, you could easily expand the example program to allow the use of multiple banks of sideways RAM, or even spare video RAM in low resolution screen modes. If you have ROM programming facilities, you could have a read-only section of the database in ROM, together with a read/write section in sideways RAM.

Aries Computers

Science Park, Milton Road, Cambridge CB4 4GD

Telephone (0223) 862614