Ultimate Ripper

POWER
COMPUTING

USER MANUAL

THE ULTIMATE CARTRIDGE

USER'S MANUAL

THE ULTIMATE CARTRIDGE TABLE OF CONTENTS

APPENDICES:

CHAPTER I

INTRODUCTION

Thank you for purchasing the ultimate cartridge from Power Computing. The cartridge incorporates a wealth of features, from disk and memory editing to capturing sound and pictures, besides more complex programming functions. We hope that you will get the most from your new toolkit, and would advise you to read this manual through and have it on hand whilst using the product.

The ultimate cartridge should be considered as a toolkit. You may be relatively new to your Atari. The Ultimate is an excellent means to become more familiar with your computer. Although you may not be able to follow all the ripper modes fully at this stage, the more you use it, the easier you will find the rippers' functions to understand, and the more you will have your Atari ST eating out of your hand! So don't worry if you're a bit baffled at this stage. The trick is to try loading in different games and demos, and to play around with the ripper until you feel confident enough to have a serious go at something!

The cartridge wlll work on your ST - whatever its configuration. Although you may not be able to use it to its fullest extent until you are familiar with your machine and the ripper, there should be no problem getting started on your own machine. lt is designed to work on ST's, STM's, STF's, STFM's and STE's - under any configuration of memory, and with any version of TOS . if the cartridge doesn't seem to work at all, we would recommend that you carefully read the trouble shooting guide at the back of this manual before contacting us.

We wish you every success with your new cartridge, and hope that you will find it a useful aid in learning more about your ST, and using that knowledge to your advantage!

CHAPTER II

A TOUR OF THE ATARI ST

This section is designed to give you a brief insight into the nuts and bolts of how your machine works inside. It will explain terms like 'hexadecimal', 'binary' and 'video RAM', that will be helpful in understanding the way to use the cartridge, later on.

First we must think about the RAM in your computer (Random Access Memory, into which you may write a program or load a game). You can tell the amount of RAM that your computer has in real terms by its name. A 520ST has 512 kilobytes of memory, and a 1040 has 1 megabyte, or 1024 Kilobytes. This is, on the face of it, rather confusing. But all you need remember is that 1 kilobyte is 1024 bytes long, as opposed to 1000.

So what is a byte? Well, like grams for measuring weight, there is a unit for measuring memory. This is called the 'bit'. Eight bits together make a byte - like 100cm and one metre. 1024 such bytes make a kilobyte.

The idea of the 'bit' is very useful. It is the base unit of any computer, and is a binary register. Let us consider number bases. We normally operate in base ten. We count from zero to nine, and then ten to nineteen, starting the process off again. Base sixteen is one of the bases that computers work in. Instead of counting zero to nine, and then starting again, in this base you would count as follows:

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

before starting again. This is called hexadecimal. Binary, also called base two, is another basis on which computers operate. In this base, two digits only are used - zero and one. The processor -called the Motorola 68000 chip - is responsible for handling the information that is written to and read from each bit of the memory. It does this through the 'address bus', and reads and writes to a succession of memory addresses - address 0 being the first byte, the second byte is address 1 and so on.

SO WHAT IS A MEMORY ADDRESS?

Consider a postman doing his rounds. Each letter he delivers in a particular road has an address on it - the number in the street. This tells him where to deliver the letter. We can consider the memory in your computer as a street, and the postman as the processor chip. The computer needs an address to deliver each letter - or piece of information to. The information that is held at each address tells the computer what to do. Posting a message to a certain address will make the computer respond in a certain way. The computer thinks of addresses in hexadecimal format, so a typical address might be $FFFF8240. (Note that the $ denotes that a number is in hexadecimal - % means that it is in binary.)

Some of the memory in your Atari is designated as 'video RAM'. When something is written to an address in video RAM, it will then be decoded by the video shifter chip, and will then appear on your screen! More information about how video RAM is handled is contained in the appendix concerning that. Once you have established a grasp on how memory addresses work, you should have little problem in getting your favourite game eating out of your hand!

CHAPTER III
GETTING STARTED

TO USE THE ULTIMATE CARTRIDGE:

Switch off your machine. Insert the cartridge into the cartridge port on your ST, with the sticker on it facing upwards. The ultimate is switched off when its switch on it is away from the computer. Put the switch in this position - it should be kept this way except for when you are actually using the cartridge to interrupt or work on a program.

You may now leave the ultimate cartridge in place, whether you intend to use it or not. It will not interfere with your machine in any way whatsoever until it is switched on.

Note that you should NEVER plug in or remove the ultimate cartridge while your machine is powered on. Damage may well result if you do not pay attention to this rule!

Now you are ready to get started. Load in the game or demo that you want to play with, and get it to the point at which you want to break in. Push the reset button on your ST, and as you hold it in, move the switch on the ultimate to the position nearest the computer. Release the reset button. Your Atari is now under the Ultimate's control! There are now two main options (others are explained in the section entitled 'Going Further'.)

The cartridge contains software that helps you to manipulate to program that you have just broken into. Just like any other piece of software, the Ultimate's software has to be loaded into memory. This happens automatically, but this unfortunately means that some of the program currently held in memory (the game you are working on) may be partly overwritten. Don't worry - you will still be able to access the whole thing - but it may take two goes!

You may either:

a) Press the F2 key. This will place the Ultimate's software in memory for you to use. This will occupy a certain amount of memory, and thus overwrite it.

b) Or, Press the Fl key. This will retrieve the overwritten part of memory and move it to another position.

You are away! You may now begin working on the game that you have successfully interrupted.

CHAPTER IV

MAIN RIPPER FUNCTIONS

PICTURE RIPPER

INTRODUCTION

Press F1 from the main Ripper screen to access the picture ripper. This function will allow you superb access to that fascinating field of your ST - the graphics. Using the picture ripper you will be able to examine the screen at the moment you pressed the reset button, with its colour palette, and, going further, even all the other graphics in memory - other screens, sprites, fonts and other uses of graphics. You will be able to understand the scrolling method for sprites, their animation, and many other exciting things about the software that you have interrupted. You will even be able to see the splicing method for sprites and screens which enable faster animation of these whilst keeping the memory load low. If you are a programmer, you will also be able to check through your programs to see that your graphics are spliced as they were first intended, in the desired places, without having to go through painful hours of debugging!

USING THE PICTURE RIPPER

Using the Picture Ripper

On the picture ripper main screen, there is a box that contains all the user adjustable parameters. These are:

— Screen address. this address refers to the memory address for the area of memory that you wish to visualise. It is always an even number, and the actual address that you enter refers to the top left hand corner of the screen.

— Number of planes. For more details on this, please refer to the appendix concerning video RAM. This parameter corresponds to the picture coding in memory. If in low resolution, this number can be from 1 to 4. In medium resolution it can be 1 or 2, and in high resolution it is always 1. This parameter is particularly useful when looking for sprites, as sprites are often coded only for the necessary planes, or logical plane by logical plane, and thus number of planes would need to be set correctly.

— Pixel width. This corresponds to the width of pixels in the picture that you want to see - and will always be a multiple of sixteen pixels (ie 16, 32, 48 etc). It is worth noting that, whilst a sprite may be very small - generally from 16 to 48 pixels - it is easy to exceed the screen width, because much graphic scenery is much larger in memorv than it is on the screen.

— Resolution. This adjusts the resolution that you want to see the picture in. Note that modification of this parameter may also change other related variables, such as logical planes (which would automatically be set to two, for example, when changing to medium resolution.)

— Planes. four parameters, each one corresponding to a specific plane. Each plane may be switched on or off by use of this function. This will have the effect of making the specific plane visible or invisible, and stars will appear on parts of the colour palette to show that the corresponding colours are no longer visible.

— Superpo. This function toggles the superimposing of planes. It may only be active under certain circumstances:

> The number of planes must be one.
> The resolution must be low.
> The pixel size must be equal or greater than the number of superimposition (1 to 4 planes, as discussed below) multiplied by the width of superimposition.

These are the only circumstances under which you may toggle the superimposing of planes on.

— Number of superpo. This defines the width in pixels of the area that you wish to superimpose. It must be a multiple of sixteen, less than 320.

— Number of planes. This sets the number of planes that you wish to superimpose.

— The large box on the right controls the colour palette. Each active colour is displayed without a * next to it, together with their corresponding hexadecimal value. You may alter these values (if the colours are active) by use of the cursor keys. The palette box is selected by pressing the F4 key. This will move the cursor from the main box to the palette section of the screen, from which you may make your modifications, and then press the F4 key again to return to the main selection box.

SPECIAL FUNCTIONS

1) Screen Display.
Press the F1 key to toggle between the menu page, and the area of video memory which you are editing (ie the picture). The screen will be in accordance to the parameters which have been set on the menu screen. Note that it is still possible to use all the keys when the menu screen is not visible - in other words you could move the cursor alongside the parameter that you wish to edit, switch screens, and then adjust the function whilst watching the result! The F3 key will not, however, operate while viewing the picture, as you are not able to save from this mode.

2) Screen Positioning.
There are several methods of changing the screen position:

a) Press the F2 key from the menu page. This will allow you to enter a new address to be edited. The address that you enter (in hexadecimal) will automatically be selected for editing and viewing.

b) If you move the cursor alongside the parameter that determines the editing address, you may increase or decrease that value by 2 bytes in either direction, plane by plane, each time you press the left or right cursor keys.

c) Press either Shift & left cursor key, or Shift & right cursor key. This will move the address by a complete screen (according to the screen size parameter which has been defined.)

d) Similarly, press either Shift & up cursor key, or Shift & down cursor key to move the screen one line at a time, according to the parameters that have been set.

3) Saving your work.
You may save the picture in DEGAS format to disk. Press the F3 key, and enter the full pathname of the filename under which you wish to save the picture. You may wish to use the filename extension P11, P12 or P13 to remind yourself in which resolution the picture has been saved. This is not a mandatory requirement, however. You will then be able to load your picture into any art package or program that recognises the DEGAS format.

4) Dumplng out the Screen to the printer.
Simply select the picture (as opposed to the menu page), and press Shift, Alternate and Help simultaneously. Ensure that your printer is set up and online first!

5) Saving to memory.
Any modifications that you make to the working screen may be quickly saved to a little memory allocated for the purpose. There are ten memory banks available to store pictures in. If you would like to save you work temporarily in memory, then just press Control and one of the keys on the numberpad (this key is then a reference for the screen that has been saved.) You may retrieve any of the memories at any point, merely by pressing the appropriate number key again. Note that there will be no notification on the screen that a 'save' has been made, or any warning to prevent you overwriting a saved memory. Also that all memories will, of course, be lost when you quit or switch off unless you select them one by one, and 'save screen' as normal.

6) The column parameter.
This function allows you to see the picture in columns - so that you may, for example, save a font in one save as opposed to three or four saves.

7) The modulo parameter.
This function allows you to recuperate a full screen picture, by adding a certain number of bytes at the end of each line.

For further detail about the picture ripper, please consult the appendix concerning 'Video RAM'.

CHAPTER V

MEMORY RIPPER

INTRODUCTION

The memory ripper is a very powerful tool. It allows you to work directly in the heart of your ST's memory. You can investigate the contents of addresses in programs, look into the state of the program in RAM, or examine the memory content in general. In addition to the standard features of a disassembler (viewing code in hexadecimal or ASCII), you may also examine the content of RAM in symbolic code (symbolics).

USING THE MEMORY RIPPER

Press F2 from the main Ripper menu to enter the memory ripper. The main memory ripper working box will appear. From left to right, the screen layout is as follows:

— The address that the pointer is at, and to which the information refers.

— The content of the address in hexadecimal.

— The content of the address in ASCII text.

If symbolic mode is selected, then the layout is:

— The current memory address.

— The content of the address in symbolics.

USING THE MEMORY RIPPER

There are several means of moving around the contents of the memory. Firstly, you can move the screen around page by page by the use of the Fl key, the space bar, or Shift & down cursor to go forwards, and the F2 key, or Shift & up cursor to move backwards. You may also move around one line at a time with the Up & Down cursor keys, and even move one byte at a time with the Right and Left cursor keys. Finally, you may select a specific address to move to, simply by pressing the F3 key and entering a memory address directly in hexadecimal. The screen will move to that location. Besides simply viewing any location of memory, there are many other functions available in the memory ripper.

SPECIAL FUNCTIONS

1) Storing your position.
In a similar fashion to the manner described in the picture ripper, there are ten slots allocated for you to store pointers for positions in memory. If you wish to save the address that you are at in RAM, then simply press control and one of the numbers on the keypad as a reference to the memory. Retrieve the contents of the memory, and return to the address held in it, by pressing the number key on its own.

2) Searching the memory.
You may make searches in the memory under a variety of formats. Press the F4 key, and then choose one - press either:

— b for a single byte
— w for a word (two bytes)
— I for a long word (four bytes)
— a for a piece of ASCII text.

You may then enter your hexadecimal number or characters, and the search will commence - starting from the address held on the screen, until:

a) The search string is found. The screen will move straight to the address at which the string is located.

b) You press the escape button. You may do this at any point, and the screen will display the address at which you stopped.

c) The search has gone as far as it is allowed. If the search goes over a permitted address zone, the screen will display the highest authorised address in memory.

If you wish to repeat your search, simply press the UNDO key, and it will recommence from the page on the display.

3) Filling Memory,
To fill a memory zone, push the F5 key. You may then enter the start address and end address for the area to be filled - either in hexadecimal or text.

4) Copying Memory.
The memory ripper allows you to copy a chunk of memory from one location to another. Press the F7 key for this option. You should then enter that start and end address for the area that you wish to copy, and then the beginning address of the area that you wish it to be moved to. It will automatically be copied directly.

5) Editing Memory.
You may edit the area of RAM that is displayed on the screen. Making sure that you are in hexadecimal mode (see function six), press F6. The cursor will appear in the box, and you may then move it around with the cursor keys. In this mode, use the TAB key to switch between zones. The position of the cursor on the screen denotes the position at which you make your modifications.

6) Symbolics toggle.
To switch between hexadecimal data and symbolic code, press the F8 key.

7) Saving an area of memory.
When you have modified an area of memory, you may well wish to save it to disk. Press F9 to save the memory. You should then enter the type of save that you require:

Press 5 to save as a source. You will need to enter the start and end address of the text zone (the zone in which the program is held), and then the end zone of the program's data. (Note that the data zone is consecutive to the text zone.) This zone will then be saved in an ASCII file that you may load into an assembler. Remember that if there is a function in the data that you have saved that makes a call to code outside the saved zone, you are likely to have problems when it comes to compiling the source that you have saved.

Press B to save data in hexadecimal. You will need to enter the start and end addresses of the zone, and a filename for it.

8) Printing a block of memory.
To start printing, ensure your printer is connected and online, and then press P. You will need to enter the start and end addresses of the zone that you wish to print, and the print will automatically be made according to the mode that you are currently in (hexadecimal or symbolic.)

CHAPTER VI

INTRODUCTION

This function of the ultimate cartridge is provided for your convenience. It will enable you to manipulate your disks as if you were still in GEM. During the course of using the ultimate cartridge, you may need to format a floppy disk - erase a file create a new folder - or any number of disk related functions. It would be inconvenient to have to drop your work and quit from the ultimate to perform these tasks, so they are included.

NB If your hard disk was not initialised at the startup time, you will be able to start it using the external TOS call function of the ultimate to run the bootup software that you use to kick the hard disk into action. You may then access it with the Disk Utility functions, just as you would from the normal GEM desktop with your partition icons.

USING THE DISK UTILITY

The main screen in this function is devoted to file lists, directory listings etc. The screen at the top shows the current pathname, together with a list of the functions. The directory (or path) that you are in at the moment is displayed when you select this mode - it may be - for example - PATH A:/*.* . Bear in mind that the word 'directory' is the same as 'folder'. The functions of this section of the ultimate are fairly straightforward:

1) Directory.
To obtain a file listing of the current directory, simply press the Fl key. A full listing of files will be produced, with the following symbols alongside to indicate the type of file that each entry represents:

Dir	— Directory.
Lab	— The 'label' that the disk was given when formatted.
Pro	— This file is read only.
Hid	— File is 'hidden' - and would not be visible from GEM.
Sys	— This is a system file.
_ _ _	—This is a standard file.

If the list exceeds the capacity of the screen, you may press Escape to pause the output, and space to recommence it.

2) Changing Directory.
Press the F2 key to change directory. You may then enter the name of the new working directory that will be used by all the ultimate's functions. Don't forget that if you wish to change drive, you will need to enter the full pathname. (eg C:/*.*.)

3) Renaming a file.
Push the F3 key to rename a file. You need then enter the old filename and the new filename, and the operation will be completed automatically.

4) Formatting a disk.
The format operation is simple - just insert the disk into drive A, write enabled, press the F4 key, and proceed as you would under GEM.

5) Erasing a file.
Simply press F5, and give the name of the file to be erased. You will be asked for confirmation, before the file is removed in a similar fashion to that under GEM.

6) Creating a directory.
Press F6 to create a new directory, and then enter its name. It will be created within the current working directory.

7) How much free space has the disk?
To find out how much free space is left on the current working drive, press the F7 key. The remaining capacity will be displayed in kilobytes.

Special Note.
If there is a problem with any of these functions, the ultimate is not able to provide very specific help towards its nature. You may well see the error message "General Error". If this is the case - think back to doing the same operation under GEM. Check that the disk is not write protected if you are trying to write to it or format it, or that you are not trying to create a directory that is already there.

CHAPTER VII

EXTERNAL CALL

This function is provided to allow you to execute external TOS programs. However, it will only function satisfactorily with programs that have no external GEM calls (pulldown menus, alert boxes and so on.) The parameters displayed on the screen are fairly self-explanatory:

> Free RAM - the remaining free memory.
>
> Start of free RAM - the start address of the free memory.
>
> Start load RAM - the starting address at which the program's basepage will be loaded. This parameter may be adjusted with the F2 key.

If your program executes successfully, you will eventually return to the same menu. Note that if you set the address for Start load RAM high enough (and that there is enough memory above this for the program to be executed), anything below it in memory will not be altered.

CHAPTER VIII

EXTERNAL LOADER

This option is to allow you to load a file or program, or a piece of text into memory, for examination or use. You may either load a complete file, or several consecutive tracks from the same side of a disk. These functions are accessed thus:

F1 — Load in a file at an address selected by you.

F2 - Load in tracks from a disk, at a selected address in memory.

CHAPTER IX

THE MUSIC RIPPER

INTRODUCTION

The music ripper is possibly the most fascinating of the ultimate's functions. It has the ability to search for music in quartet format, as digitised data, as soundtracker modules, or as chip sound itself. (More detail on these terms may be found in the appendix concerning sound and the ultimate.)

USING THE SOUND RIPPER

The sound ripper is composed of two sections. Initially one starts in the 'chip music' section - from here you may select the function for searching for digitised music. To search for chip sound:

Press the Fl key. This will allow you to search between two defined memory addresses for chip sound. This function recognises various formats of soundtracker modules, as well as most chip sound. However, the process of searching memory for chip music is far from simple, and one cannot expect a success rate much over 50% for recent programs. However, the ultimate does undeniably provide a highly simple user interface for efficiently scanning the memory for locating most chip sound. Chip sound that the cartridge successfully finds may be played directly or saved to disk, either as an executable file, runable directly from desktop, or as data. Rarely, the cartridge may inform you that an area of chip sound has been located, yet when you listen to it there is silence! This is due to the awkward nature of the way in which the data is stored. In this case, you should restart your search, giving a larger 'end' address. Once you have entered your start and end addresses, and any chip sound has been located, it may be saved by pressing:

F2 - Saving chip sound to disk. Once the cartridge has located a piece of chip sound, the start and end addresses of that chip sound will be displayed on the screen. If you are intending to save a piece of chip sound that you have just found, it is sensible to rerun the search before saving, because when the sound is initially found, a number of settings are established which may be changed or removed when the sound is listened to.

SAVING 'QUARTET' CHIP SOUND

A slightly more complex procedure is required to save chip sound which is stored in the quartet format:

a) Save the area of memory as usual, from the start to the end address, with the file extension .SET (eg music.set)

b) Save the area of memory again, giving the start address as shown on the screen, and adding 20 kilobytes on to this as the end address. Give this file the same name, but the extension .4V (eg music.4V)

USING THE DIGITISED SOUND RIPPER

This allows you physically to visualise the computer's RAM, and from this visualisation, to find and immediately play back pieces of digitised sound. The process of digitising sound is explained in the appendix. In this section, you may explore the computer's memory, extract digitised music7 and save it to disk. It is a subsection to the main sound ripper menu. accessed by pressing the F3 key.

The main screen gives a visual perspective of the memory under examination, and a menu bar to display functions and parameters. The following options are available:

Fl - Enter the start address of the area of memory to visualise so that you may search for digitised sound.

F2 - Enter the end address of the area to search. You may make fine adjustments to these values by using the cursor keys.

F3 - Alter the frequency at which any sample you find is played back. It may be adjusted between the values of 2.5 Khz and 25 Khz. Play with this value to alter the speed of playback, and thus find out what the sample is, and how it should sound!

F4 - Play back the music located between the start and end addresses that you have defined. You may stop the playback by pressing escape, and the address currently being played is shown at the bottom of the screen (except for at playback frequency 25 Khz), so that you may locate exactly where the sample is held in the displayed memory.

F6 - Specify whether the music is signed or not. You should try this function if the playback of the sample appears to suffer from interference.

F5 - Save a chunk of memory. This may be done either as data to be listened to later (eg from ST-REPLAY), or as an executable file. The ultimate will create a program to playback the chosen sample at the frequency and other parameters defined when it is saved.

SPECIAL POINTS

If you save music to disk in executable or data format and nothing appears to happen at playback - either no music was found in the first place - make sure you listen to the sound before saving it to disk, or the chunk of memory saved was too large. You should aim not to over estimate starting and ending addresses, as an end address located way after the actual end of the music will cause the playback to fail.

If you are a programmer:

To stop playback of digitised music. you need only stop the A timer, using the #$1A XBIOS function (or Jdisint function) - in assembly:

```
MOVE.W #13,-(A7)    : timer A
MOVE.W #$1A-(A7)    : Jdisint function
TRAP #14            : Xbios call
ADDQ.L #4,A7        : Stack restoration
```

ATARI STE

If you have an Atari STE, you will need to use the function F7 to toggle into STE mode, from the sample ripper. In STE mode, there are four available frequencies - 6.25 Khz, 12.5 Khz, 25 Khz or 50 Khz.

You can control the STE sound by pressing the F6 key:

General volume [0-40]
Right volume [0-20]
Left volume [0-20]
Treble [0-12]
Bass [0-12]
Stereo [On/Off]

CHAPTER X

DISK RIPPER

INTRODUCTION

This function of the cartridge is very powerful. It allows you access to your disks and disk drives at an exceptionally low level. It allows you the freedom of access to your disks that the memory ripper allows to your RAM. You can examine the contents of a diskette in hexadecimal or symbolic code, check the state of sectors (including the boot sectors of disks), and read information directly from each track of the disk.

USING THE DISK RIPPER

The layout of this part of the ultimate is the same as the memory ripper. Further to this, however, there are two additional columns of information on the main display - the length of the block just read, and the error code received after reading.

When you enter the disk editor, a menu box will appear, showing all the diskette defaults (2 sides, 9 sectors, 80 tracks.) You may then read the disk information (or boot sector) from the current disk, by pressing the Fl key. Instead, you may modify the disk information by hand, by moving to the parameter that you wish to alter, with the cursor keys, and then changing its value by pressing either the left or right cursor buttons. The current information is that displayed under the selection menu - current drive, track, sector and side.

READING SECTORS

To read the current sector in, press the Fl key. This sector will then be read and its contents will be displayed in the viewing box, together with the length of the sector and the message obtained from reading it in (this should read 'SECTOR OK' if there are no disk errors). If you change the information at the top of the screen, (say to a different sector), the disk will not automatically be read in. You must actually press the Fl key to read the information from the disk. To move around the disk sector by sector, use the plus and minus keys from the number pad - this will read information sequentially from the disk, according to the setup parameters that you entered at the beginning. You may also move around the disk rapidly, track by track, by holding down control and pressing the + and - keys.

WRITING SECTORS

If you have modified a sector, and wish to write it to disk, press the F2 key. The sector will then be written at the current position on the disk (following confirmation). You may thus use this function to copy a sector from place to place, by reading a sector in, adjusting the position on the disk, and then saving it again.

SEARCHING A DISK

Enter this mode by pressing the F3 key. You may make a search either in hexadecimal or in text for a string to be looked for and then an optional replacement string. The search will thus commence, continuing until you press the escape key to terminate the search, or the string is found on the disk. In this case, a menu bar will appear, with the following options available:

— Replace the search string with the replacement string (if you entered one — otherwise this section is unavailable.)

— Continue the search.

— Stop searching and go to the string that you found.

— Stop searching at this point.

READING TRACK INFORMATION

Press the F4 key to read the current track information from the disk. The values that are read in are:

— The track found in the ID field.
— The side found in the ID field.
— The number of sectors and track.
— The sectors' sizes in bytes.
— The ID field checksum.
— The ID field checksum test.
— The ID field encounter test.
— The sector reading test.

Further information on this function may be found in the appropriate appendix.

READING IN A TRACK

In a similar way to Fl reads in a sector, F5 allows you to read in the current track. The functions are otherwise identical.

COPYING

The copy function allows you to copy from the disk buffer into memory, or vice versa, simply by entering the source or destination address in memory. You are advised not to save a buffer below memory address $17D00, because the system and ultimate cartridge have this memory reserved. You will be asked for confirmation if you try to save to this area of memory.

DISK INFORMATION

This function, accessible via F6, gives you access to the menu box of initial information that was displayed when you first started the disk editor.

EDITING

This function works in a similar fashion to the edit function of the memory ripper. It operates only in sector reading mode, and is used by pressing F7.

SYMBOLICS

The F8 key toggles between a display in hexadecimal, and a display in symbolic code.

PRINTING

To print the current sector or track, ensure that your printer is ready, online, and connected. Press the P key to start printing.

ERROR MESSAGES

The following error messages may be obtained whilst reading in sectors:

SECTOR OK — The sector was read in fine.

SECTOR NOT FOUND — You have tried to read in a nonexistent sector. Test out the disk's information on this track to check the number of sectors present.

PROTECTED DISK — The disk is write protected, and you tried to write to it.

DISK ERROR — An error occurred during reading or writing.

BAD DAM — After reading in, this error may happen if the sector's data mark or the ID field of the sector has not been found.

BAD DAC — The data field checksum is wrong.

BAD IDC — The ID field checksum is wrong.

BAD DAR — There was no reaction to a data request.

DAM $F8 — The data mark is erased.

CHAPTER XI

INFO RIPPER

This function displays information about the designers and copyright of the ripper software and hardware, that you should read.

CHAPTER XII

QUITTING FROM THE ULTIMATE

The F10 key will return you to GEM and the ultimate will no longer control your machine. You are advised to return the switch to the position away from the computer, and then to reboot the machine.

APPENDIX A

MORE ABOUT MEMORY

The atari has RAM, ROM, TOS and GEM! We will briefly have a look at each of these terms:

ROM - Contains the information that makes your Atari ST and Atari ST. You may not write to, only read information from this area of memory, and hence it is termed Read Only Memory.

GEM and TOS - TOS is 'The Operating System'. GEM is the graphical environment under which your Atari works. It is, if you like, the software which your computer runs as soon as you switch it on. It means that your disk drives appear on the screen as icons, and provides the formatting dialogue boxes for you, and so on. When you switch your machine on, TOS starts up, and then GEM runs. Each of these uses a little bit of memory from your ST.

RAM - This is 'Random Access Memory' - memory that is yours to read from or write to, and you are free to hack at this memory with the ultimate. Note, however, that, as stated above, a small amount of this memory is required for TOS and GEM. The amount of memory used by these depends on your version of TOS, but it will always occupy the very beginning of the memory. You are advised not to write from the ultimate into this area of your RAM, as this is almost certain to cause the computer to crash!

APPENDIX B

MORE ABOUT VIDEO MEMORY

Like any computer, the ST has a video output. The video out will be displayed on your screen after being decoded, and the source of this is part of the video memory called video RAM. On the ST, this video RAM has a standard size of 32000 bytes, and can be at any point in memory, as long as its address is a multiple of 256. The video shifter chip decodes the contents of the video RAM and sends it to the screen, in a format according to the resolution in which you are working - high, medium, or low. Picture coding is also different according to resolution, as the number of logical planes depends on this. This is a colour coding device, as the video shifter chip has to decode colours according to the resolution that you are currently in:

Low res. - 4 logical planes, 320 x 200 pixels, in 16 colours.
Medium res. - 2 logical planes, 640 x 200 pixels, in 2 colours.
High res. - 1 logical plane, 640 x 400 pixels, in 1 colour.

Coding for high resolution is simple. A pixel can only be one colour, and thus a single bit (one logical plane) is sufficient to code the state of a pixel - a 0 bit corresponds to a switched off pixel, and a 1 bit to a switched on pixel.

However, the coding for medium resolution is somewhat more complex. A pixel can have four different colours, and thus more than one on/off bit is required. Two bits (two logical planes) are required. Let us call these two bits bita and bitb:

Value of the two bits	Resulting colour
%0(bitb = 0, bita = 0)	colour 0
%01 (bitb = 0, bita = 1)	colour 1
%10(bitb = 1, bita = 0)	colour 2
%11(bitb = 1, bita = 1)	colour 3

To make things slightly more awkward, these two bits are not consecutive in memory, the coding is done in groups of 16 pixels, and thus this set is made of 32 bits (two words), divided into two subgroups of 16 bits (one word) that we will call a and b. The subgroup a is made of all the bita bits, and the subset b of all the bitb bits of the 16 pixels. Thus, if we were, for example, to code the first pixel to colour two:

bita = O and bitb = 1 — thus the 32 bits would be:

%Oxxxxxxxxxxxxxxx	%lxxxxxxxxxxxxxxx
Group A.	Group B.

(where x is the state of any one bit.)

For low resolution, the same principle of coding is apparent for the four logical planes. Since each pixel may have sixteen possible colours, coding must by via four bits (bita, bitb, bitc, and bitd.)

Value of the 4 bits.	Resulting Colour.
%0000	Colour O
%0001	Colour 1
%0010	Colour 2
%0011	Colour 3
%0100	Colour 4
%0101	Colour 5
%0110	Colour 6
%0111	Colour 7
%1000	Colour 8
%1001	Colour 9
%1010	Colour 10
%1011	Colour 11
%1100	Colour 12
%1101	Colour 13
%1110	Colour 14
%1111	Colour 15

Thus if, for example, we wished to code the first pixel to colour 10:

bita = 0, bitb = 1, bitc = 0, bitd = 1.

Oxxxxxxxxxxxxxxx	%1xxxxxxxxxxxxxxx	%Oxxxxxxxxxxxxxxx
Group a.	Group b.	Group c.

%1 xxxxxxxxxxxxxxx
Group d.

So a low resolution picture is always coded by groups of four words corresponding to the bit's grouping, and thus the picture can use sixteen colours from the colour palette. However, it is not mandatory that a sprite, for example, should actually use all sixteen colours. Suppose it only uses eight of the colours available — say the first eight colours. In this case, only three bits are necessary to code each pixel, as the fourth bit will automatically be set to zero:

Value of the 4 bits	Resulting colour
%0x	Colour 0
%001x	Colour 1
%010x	Colour 2
%011x	Colour 3
%100x	Colour 4
%101x	Colour 5
%110x	Colour 6
%lllx	Colour 7

To show such a sprite on the screen, the video shifter decoding must use the usual four logical planes, (group d will be set to zero), but it would be pointless to store the group d in memory as its value must be zero, and so only the three groups with specific values are stored, giving the following structure in memory:

groupa 1, groupb 1, groupc 1, groupa 2, groupb 2, groupc 2....

If we are thus storing 25% less information, we have memory capacity for many more sprites. It is important, thus, that you make searches for sprites coded in three logical planes only.

ADDRESSES USED BY THE VIDEO

SHIFTER

Colour Palette.	Resulting colour.
$FFFF8240	Colour O
$FFFF8242	Colour 1
$FFFF8244	Colour 2
$FFFF8246	Colour 3
$FFFF8248	Colour 4
$FFFF824A	Colour 5
$FFFF824C	Colour 6
$FFFF824E	Colour 7
$FFFF8250	Colour 8
$FFFF8252	Colour 9
$FFFF8254	Colour 10
$FFFF8256	Colour 11
$FFFF8258	Colour 12
$FFFF825A	Colour 13
$FFFF825C	Colour 14
$FFFF825E	Colour 15

RESOLUTION ADDRESS

This address signals the resolution, and thus the coding system with which the video RAM is decoded:

$FFFF8260

contains: %xxxxxxxxxxxxxx00 — Low resolution.
%xxxxxxxxxxxxxx01 — Medium resolution.
%xxxxxxxxxxxxxx10 — High resolution.

(Where x can be either 1 or 0).

SYNCHRONISATION ADDRESS

This address contains the synchronisation mode of the screen. The European synchronisation is 50Hz, whilst in the US it would be 60Hz:

$FFFF820A

contains: %xxxxxx00 — 60 Hz
%xxxxxx10 — 50 Hz
%xxxxxxx1 — External synchronisation.

VIDEO RAM ADDRESS

This address denotes the start of your video RAM, read by the video shifter once every screen frame. This address is always a multiple of 256 (except in the case of the STE), and thus there is no 'lower' part:

$FFFF8201 — High part of video memory.
$FFFF8203 — Medium part of video memory.

VIDEO POINTER ADDRESS

This address contains the address of the pixel actually shown on the screen:

$FFFF8205 — High part of the video pointer.
$FFFF8207 — Medium part of the video pointer.
$FFFF8209 — Low part of the video pointer.

SUMMARY

It is clear that by changing certain addresses we can obtain visual effects on the screen — for example, by modifying the address of the video memory, it is possible to obtain certain animations corresponding to the scrolling of that memory. Or, by modifying the resolution we can have half the screen in medium resolution, and the other half in low (or neochrome). Understanding the coding system by which the video RAM is translated on to your screen is important in order to get the most from the Image Ripper section of the Ultimate.

APPENDIX C

MORE ABOUT SOUND

Chip Sound: Sound produced by the Atari's sound chip.
Digitised sound: A sound which has been sampled and stored.

THE SOUND CHIP

The Atari's sound chip is quite versatile, and when well programmed, can produce a good variety of different sounds. It can produce three different sounds at the same time from three dedicated sound channels, A, B and C. It is a complex process to program this chip well, and it is sufficient to say that its possibilities have now essentially been outdated. It does, however, require very little memory to control the chip well (from about 3Kb to 15Kb), and will by no means overload the machine — it is capable of doing many other things simultaneously.

DIGITISED SOUND

The process of playing digitised sound is very different. When you sample a sound (for example with the ST REPLAY), the computer calculates a number value from the pitch and volume of the sound many many times a second. When these numbers are reprocessed, the sound is reproduced.

The problem is that, in order to reproduce a sound at a reasonable quality, many samples have to be taken a second — up 44000 times per second (44 Khz) to produce compact disk quality. This means that the memory requirement to store such a sample is very high. Suppose you sampled a 20 second piece of music 10000 times a second — ie at a frequency of 10 Khz. You will be reading data 10000 times a second for 20 seconds, and will thus require 200000 bytes to store the information. Play back is then simple merely being a case of converting one piece of data every ten thousandth of a second, and at this degree of speed, the drain on the Atari's system is very substantial, and it is hard to do anything else at the same time!

The advantages are great, however, as the scope of sounds that may be sampled is endless — you may sample anything for as long as memory permits, and the playback will sound exactly like the original music or noise. Playback requires no extra hardware other than your ST.

APPENDIX D

MORE ABOUT THE DISK DRIVE

What actually happens when you 'format' a disk to 720Kb? A diskette is divided into circular rings called tracks, and formatting a disk makes these tracks ready to store the information that you will write to them. It divides each track into 9, 10 or 11 equal parts with a particular structure. These chunks are called sectors, and generally have a capacity of 512 bytes.

THE STRUCTURE OF A TRACK

gap 1
gap 2
synchronisation
ID—AM (index address mark)
track number
side number
sector number
sector length
gap 3
synchronisation
DAM (data mark)
sector data
DATA CRC (checksum of data field)
gap 4

(Repeated — according to how many sectors there are)

gap 5

A gap is a field which separates the different pieces of information held on a track, containing no useful data, but present to give the disk drive a certain time to identify its position in the track.

Gap 1 — or the pre-track gap, enables the drive to position the head correctly on the start of the data on the track. On a disk with a standard format it is about 60 bytes long — each byte having a value of $4e.

Gap 2 — enables the head to be positioned on the first piece of the data of a sector. Its length is 12 bytes with the value of 0.

Synchronisation is a succession of 3 bytes with the value $FE, indicating the beginning of an index field to the controller.

The ID-AM is one byte long, and always has a value of $FE, indicating the beginning of an index field to the controller.

Track Number — indicates which track the head is currently on. One byte in length.

Side Number — indicates which side the head is working on. This may be either 0 or 1, and is one byte long.

Sector Number — shows the sector that follows this index field. As it is one byte long, a sector can be numbered between 0 and 255 ($0 to $FF).

Sector Size — this is coded on one byte, and can take four different values:

0 — 128 bytes long
1 — 256 bytes long
1 — 512 bytes long
1 — 1024 bytes long

ID CRC is two bytes long, and enables the controller to check that the index field information is correct.

Gap 3 — Enables the separation of the index field and the data field, and is divided into two parts:

22 bytes at $4E
12 bytes at $00

DAM — indicates the actual beginning of the sector — its value is $FB for a normal DAM, or $F8 for an erased DAM. The sector's data has a variable length, in relation to the size definition in the index field.

Gap 4 — The end of the sector. This gap is generally 40 bytes long, each byte having the value of $4E.

Gap 5 — The end of the track. This has a very variable size, from 30 to 700 bytes according to the size and number of sectors.

This information will help us to understand why it is necessary to format a disk before writing to it — and why it is not possible to format and write simultaneously. This is because while a disk is being formatted, the controller reads certain bytes as being control bytes, and if you wished to store such a byte it would be decoded by the controller in mistake for an information byte, rather than being written to the disk. However, once the format operation is completed, you may write any data field you wish — including those used as control bytes.

These are the control bytes:

$F5 — Synchronisation byte, translated to $A1 on the disk, enables zeroing of the CRC register.

$F6 — Synchronisation byte, translated to SC2 on the disk does not zero the CRC register.

$F7 — Calculation byte of the CRC, translated into two bytes on the disk, resulting from the CRC calculation.

$F8 — Tells the controller to write an erased data mark.

$FB — A normal data mark.

$FE — An index address data mark.

$F9, $F10, $FC, $FD, $FF — A data mark (not used on the ST.)

FILE MANAGEMENT

Once the disk has been formatted, the ST needs to be able to tell where to store any files you write to it, and by what means. There are thus some areas on the disk reserved for system information to be written, called the boot sector, the directory, and the file allocation table (or FAT).

THE BOOT SECTOR

This is always at the beginning of the disk — track 0, side 0, sector 0 or 1. The boot sector is used to store the disk information, and possibly a small boot-up program. There is a series of parameters held in this sector, detailed below:

No Bytes.	Information Stored.
2	Branching, in case it is an executable boot sector.
6	Fillers.
4	The serial number.
2	Inverted bytes for the number of sectors per cluster.
2	Inverted bytes for the number of reserved sectors.
2	Number of fats.
2	Number of files per directory.
2	Number of sectors on the disk.
2	Medium description.
2	Number of sectors per fat.
2	Number of sectors per track.
2	Number of sides on the disk.

THE DIRECTORY

This is located on track 1, sector 3, and contains all the necessary file information, for each file stored on the disk. 32 bytes of information is stored in each case:-

No Bytes.	Information Stored.
8	File name.
3	File extension.
1	File attributes.
10	Reserved bytes.
2	Modification time.
2	Modification date.
2	First cluster of the file.
4	Size of the file.

The first byte of the filename also contains certain information:

If it is zero, then the field has never been used, and is free.

If it is $E5, then the file has been erased, and the area is free to be used again.

THE FAT (FILE ALLOCATION TABLE)

This allows the system to account for all the clusters of a file (a cluster is generally made of two consecutive sectors). All the data in this table is encoded thus in 12 or 16 bits:

0	—	The cluster is free.
$FF7	—	The cluster is corrupted.
$FFF	—	The last cluster on the list of clusters referring to the current file.

Any other value indicates that the subsequent cluster is the next cluster of the file.

APPENDIX E

FURTHER TIPS FOR STARTING THE ULTIMATE

USING THE F1 KEY TO GET STARTED

For the ultimate to work, it is necessary that the RAM between the addresses $4 and $17D00 is overwritten. By use of the Fl key, however, it is possible to save the contents of this area elsewhere. This area will be moved either to $60000, or $80000 on an ST with more than 512K RAM.

WHERE IS THE ULTIMATE SCREEN HELD?

The screen address of the Ripper is from addresses $10000 to $17D00, and its BSS section is below this. You are advised not to modify RAM below these addresses to avoid crashing the machine.

CAN I JUST SWITCH THE CARTRIDGE OFF?

No. This will almost certainly cause the machine to crash immediately.

STARTING THE ULTIMATE WITH THE F7 KEY

This option takes you into 'trainer' mode. Further details on this are covered in the the appendix concerning the Ring Interruption Lead.

STARTING THE ULTIMATE WITH THE F9 KEY

This function is only useful for programmers whilst debugging. This allows you to read the content of address $30 at reset time, and to jump to this address.

USING THE F10 KEY

If you left the cartridge on when you switched on your machine, the F10 key will allow you to perform a normal reset. You are, however, advised to switch it off and reboot the machine each time you finish using it.

APPENDIX F

KEYBOARD SHORTCUTS

Ctrl 4 c —	When these keys are pressed together, a calculator box is provided from within the ultimate for small arithmetic functions involving two arguments.

From any menu, you may switch to another menu, bypassing the main Ripper menu, by the use of the following keys:

Shift F1 — Picture Ripper
Shift F2 — Memory Ripper
Shift F3 — Disk Utility
Shift F4 — External Call
Shift F5 — External Loader
Shift F6 — Music Ripper
Shift F7 — Disk Ripper
Shift F8 — Ripper Information

APPENDIX G

THE RING INTERRUPTION LEAD

This function allows you to stop a program and then restart it after hacking it, and also to attempt a search for an infinite lives string. To enable this, you must first construct the ring interruption lead.

For this cable, you will need a female RS232 connector (a 25 pin plug), a piece of wire, and a bell push. Cut the wire into two pieces, and connect on end of each to the bell push on either side of it. Then connect the two ends to pins 20 and 22 or the connector, so make a switching device. Plug this connector into the modem port of your ST (the one labelled with a telephone receiver.)

With this cable installed, run the program which you wish to interrupt, and break into it with the ultimate as usual, using the F7 key once you have pressed the reset button. You may now halt the program at any point by pressing the bell push on your lead. From this point, a menu bar will appear, giving you the option to:

F10 — Continue the program from the point at which it was broken into.

F1 — Enter the search and replace function. You must then enter:

a) A string to be searched for, in hexadecimal.
b) A string to replace it with, if it is found.

The search will then commence, until it either fails, and stops automatically, or the string is found. At this stage, the address at which the string has been found is displayed, and the following options are available:

a) Change the search string, with the Fl key.
b) Continue searching, with the F2 key.
c) Conclude the search, with the F10 key.

You may then continue with the program, at the point at which you left off.

APPENDIX H
TROUBLE SHOOTING GUIDE

The ultimate cartridge is a product which has taken the utmost care in development and production. If the answer to your question is not in this manual, it probably means that what you want to do can only be achieved by experimenting with the ultimate. It is a toolbox, and the more you use it, the easier it will become to use.

Every ultimate that leaves our premises is thoroughly tested before it is packed. There is very little in it that can possibly go wrong. If you can't get the cartridge to work at all, or even in part, we would ask that you reread the manual carefully, in full, and then take the following steps:

Check that you have inserted the cartridge with the sticker facing upwards.

Check that there is a disk present in drive A at all times.

Check that, when you switch your machine on, the switch on the ultimate is positioned away from the machine. Then carefully follow the instructions for breaking into a program, using the GEM desktop as if it were a program to break into, rather than attempting to load and break into a game.

Important! Switch off your machine, and remove the ultimate cartridge. Clean thoroughly the pins on the ultimate and on the cartridge port of your computer with alcohol (eg with methylated spirits). Allow them to dry completely before retrying. This will almost 100% certainly be the solution to any problem that you have with the ultimate - so please try it!

If none of the above has proved successful, please contact our technical support lines on the number advertised in the current magazine. Please be patient! Our lines are busy. If you have a modem, you may leave a request for technical support by dialling the following number and leaving your details : (0203) 638780.

Updates to the Ripper will be available in the future from Power Computing, and these will be widely advertised in the press, to ensure that you are fully informed. We hope that you find the ultimate to be both educational and enjoyable, and that you will find its use to be simple and problem free.

POWER COMPUTING LTD

Unit 8, Railton Road, Woburn Road Industrial Estate, Kempston, Bedford MK42 7PN

TEL: (0234) 843388 (Five Lines) • FAX: (0234) 840234