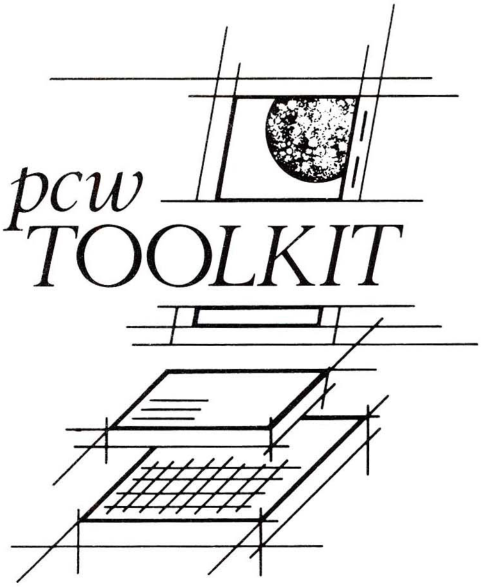


pcw TOOLKIT

MOONSTONE

COMPUTING

PCW-ToolKit

Version 2

User Guide

Written by Colin Foster

Published by Moonstone Computing © 1989

Contents

INTRODUCTION

Welcome to the User Guide for Moonstone Computing's **PCW-ToolKit** program. This book is designed to explain just what the program does, and to help you get the most out of it.

Part One deals with copying of your master disc to create a working copy, followed by a general but comprehensive guide to the concepts involved in disc formats, and the CP/M operating system and Locoscript as they affects us.

Part Two is a First Aid Kit! If you are reading this because you have just ERAsed a vital file, or because The Most Important Disc In The World has just let you down with a "No Data" or "Missing Address Mark" error at a crucial moment, then start here! We cover the main problems that drive people to contemplate suicide, and provide a step-by-step guide to saving the day!

Part Three describes in detail the various functions ToolKit provides, and provides a step-by-step guide to their uses.

Part Four is a comprehensive technical reference section for those of you who require such information, or are interested in learning more about how discs work, containing in-depth information on several topics which were covered more generally in the main part of the manual. This includes the PCW's Disc Directory structure and a full explanation of the various disc errors which can occur, and their causes.

Throughout this manual, all topics will be presented in as simple and straightforward a manner as possible. ToolKit is designed to be simple and friendly to use, and you do ***not*** need any detailed knowledge of CP/M before you can edit data or resurrect damaged files. However, some of the more difficult tasks ToolKit can help you carry out (such as rebuilding corrupted directories, for example) ***do*** require some knowledge of the way CP/M stores files on discs. This is covered in some detail in Part Four.

Copying ToolKit

Please read the Copyright notice at the beginning of this manual. ToolKit is ***not*** copy-protected, and you ***are*** allowed (and we would encourage you) to make a copy of your ToolKit distribution disc, so long as this is used soley as your own working copy and the original is kept out of use, in a safe place. (The procedure to copy the disc is described in Part One).

You are ***not***, however, allowed to make a copy to give to someone else, even if it is "just for them to try", or to use copies of the same program on more than one computer. Not only is any such copy ***illegal***, but each sale we lose in this way simply makes it less likely that we will upgrade or develop this program further - in other words, ***you*** will lose out too.

You paid for ToolKit - if someone else wants a copy, tell them to do the same!

Registration and Customer Support

Please complete and post the Registration Card included in your PCW-ToolKit package as soon as possible. This allows us to send you a Registration Number, which we will require if you ever phone or write to us for assistance. If you have not registered, we will ***not*** be able to help you.

Registration also ensures that you will be eligible for any future upgrades to the program.

We are always willing to offer all reasonable support and assistance to our customers; if you have a problem or a special requirement please contact us. If you write, we would appreciate a stamped S.A.E for our reply; while if you have an urgent problem then please try to phone between 3pm and 5pm weekdays, when technical staff will generally be available to help you.

So what does ToolKit do?

Basically, the program is designed to allow you to examine and alter data on any PCW disc, as well as copy part or all of damaged or corrupted discs and repair the damage. This means you can recover most or all of important data from such a disc, saving much retyping or worse. ToolKit also provides the means to Un-ERAse a file - how often have you wished for that?!

ToolKit is designed to be simple and intuitive to use. At all times, a menu displays the options available to you - you don't have to memorise the manual to know what you can do when, as with some software! Also, you give ToolKit commands by using either the first letter of the command, or a special key such as **COPY** or **PASTE**, and the **CAN**cel and **EXIT** keys do what they say, just as in Locoscript. You ***don't*** need to remember bizarre or obscure commands like "**ALT + f2** means quit", as you have to with some programs!

Cursor Keys are used to browse around the Tracks and Sectors which make up a disc, and sectors can be read, their contents altered and re-written to disc with just a few simple commands.

ToolKit's **COPY** function will do what other programs such as DISCKIT won't - copy a single track or file, or even an entire disc, ***regardless*** of any errors or damage. Normal programs, such as Locoscript and CP/M's DISCKIT and PIP, abort reading data whenever they detect an error, so if you have a bad sector near the start of a file, you'd find it impossible to get at the remaining (and perfectly healthy!) data which follows it. ToolKit, on the other hand, ***will*** keep going to recover as much as possible from a file or disc, as well as providing you with the facilities to rebuild or repair the damaged data.

PART ONE

The Basics

Copying your Master Disc

Your copy of **PCW-ToolKit** is supplied on a 40-track 3" Amstrad CF2 format disc. This is the type of disc used by the PCW8256 and PCW8512 in Drive A, and only one side of the disc can be read by the computer (the side facing towards the screen). However, like a cassette tape, these discs can be turned round so as the disc drive can use the other side.

The disc drive fitted to the PCW9512 is a higher-capacity double-sided unit, the same as is used as Drive B: on a PCW8512. This can, however, read your master disc just the same.

For safety, identical copies of PCW-ToolKit are recorded on both sides of your master disc. This means that should the computer have any difficulty reading the first side of the disc, you can turn it over and read the other side instead.

If possible, you should never use routinely the original master copies of any software you have. (This includes Locoscript and CP/M.) If these are damaged in use, it will be difficult (and possibly expensive) to get them replaced. Instead, you should always make a ***working copy*** of a master disc and use that instead. Some commercial software is copy-protected to prevent you doing this, but PCW-ToolKit is not - we prefer to trust our customers to respect our copyright rather than treat you all as potential criminals the way some companies do!

PCW8256 and 8512 owners can use **DISCKIT** to make a copy of their master disc if they wish; however, for reasons we'll explain later on, this is ***not*** possible if you have a PCW9512. So, to avoid confusion, we'll describe a way to copy the disc which everyone can use!

First, you must boot up CP/M if you have not already done so. This is on the other side of your Locoscript master disc (if you have a

PCW8256 or 8512) or is on a separate disc if you have a PCW9512. You start CP/M in exactly the same way as you would start Loco-script, simply by inserting the disc after you have either turned the computer on or reset it by pressing **SHIFT+EXTRA+EXIT**.

You will see CP/M sign on with a message describing the number of disc drives fitted to your computer, the amount of memory available for use as a RAM disc, and details of any add-ons (such as a serial interface) you have attached. It will then display a prompt -

A>

and wait for you to type a command.

If you do not have a blank, formatted disc to hand, then you must use **DISCKIT** to create one. Type

A>**disckit <RETURN>**

(Note that when you see **<RETURN>**, we mean "press the **RETURN** key", not "type <-R-E-T-U-R-N->"! Also, CP/M doesn't care whether you type commands in upper case, lower case or a mixture of them both.)

Follow DISCKIT's prompts to format a disc in Drive A:. This is the disc we will use to hold the working copy of your master disc.

Once you have a formatted disc to hand, make sure your CP/M disc is in Drive A: and type

A>**pip <RETURN>**

PIP is the name of the CP/M command used to copy files from one disc to another, and displays another prompt of its own -

`*`

Next, place your **PCW-ToolKit** master disc into Drive A: and type

`*m:=a:*.*[v]` <RETURN>

This command tells PIP to make a duplicate copy of all the files present on the master disc, and put the copies into the RAM disc (called Drive

M:). The **[v]** is an ***option*** which tells **PIP** to verify all the copies it makes against the originals, to make sure that they are identical and no errors have occurred.

When PIP has copied the files and is waiting for another command, remove the ToolKit master disc from Drive A: and replace it with your blank disc.

Now type

***a:=m:*.*[v] <RETURN>**

This does exactly the opposite of the last command - it tells PIP to make a copy of all the files in Drive M: and put the copies onto the blank disc in Drive A:

When PIP has finished, remove the disc from Drive A:, close the write-protect tab to avoid accidentally overwriting any of the information on it at a later date, and label it as your Working Copy of PCW-ToolKit.

Put your master disc away in a safe place - use it only to make another working copy should something happen to this one.

Additional information - the READ.ME file

If your disc has a file called **READ.ME** on it, then this contains supplementary information on your version of PCW-ToolKit. You can read this by typing

A>**type read.me**

or you can view and print it out from Locoscript by first loading it into an empty document as a Text Block.

In the remainder of Part One, we'll look at how discs work, and explain the terminology used in ToolKit.

How do discs work ?

A floppy disc is simply a disc of plastic (3" in diameter, in our case), coated in the same sort of magnetic material as a cassette tape. Before a computer can use a disc, it must be ***formatted*** - more of that in a moment - a process which writes a preset data pattern all over the disc, dividing it up into areas which the computer can then address by number. These areas are called ***sectors***, and on the PCW they can each hold ***512 bytes*** of data (0.5 Kbytes).

On a record, the music is engraved serially in one long line, which spirals in from the outside of the record to the centre to pack as much information in as possible. Computers could obviously use a similar system to store sectors on a magnetic disc; however, unlike a record, we rarely want to "listen" to the first half of the disc to get to some information stored in the middle!

Also, without some sort of complicated indexing system and drive mechanism (such as that used on music CDs), the computer would not be able to "goto" a particular sector if they were recorded serially in a spiral.

Instead, a slightly different system is used - the disc is split up into a number of concentric circles of data, called ***tracks***, rather than a single long spiral.

Formatting

Within each track, the formatting operation writes a rather complex pattern involving a header block to identify the start of the track followed by a number of (empty) sectors. Each sector on a track also has a "hidden" block of information attached to it, which contains things like the sector's unique ID number and a complex checksum (called a ***Cyclic Redundancy Checksum***, or **CRC**) used to ensure the integrity of the data which is stored in the sector and warn the computer if it becomes corrupted.

Each sector on a track has an ID number assigned to it - the first sector on each track has number 1, and the rest of the sectors number consecutively from this. Thus we can identify a sector uniquely by specifying which side of the disc and which track it is on, and which sector number it is within that track.

Note that once a disc has been formatted to be a "Double Density" CF2DD, it ***cannot*** be then used as a CF2 disc in a PCW8256 or 8512's Drive A: - it just isn't possible for a 40- track, single-sided disc drive to read a disc with two sides and 80 tracks of data per side!

However, the reverse IS possible to some extent, which is the reason the higher capacity drives can read CF2 discs (though for technical reasons they can't ***write*** to them, which is the reason that DISCKIT can't be used to copy a 40-track CF2 disc on a PCW9512).

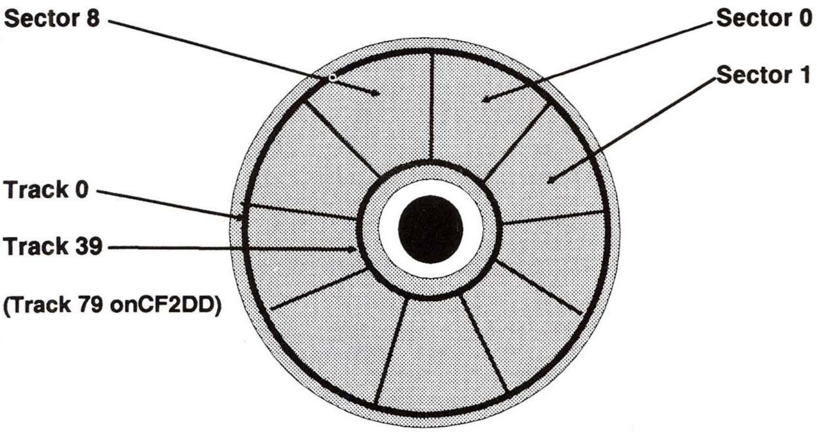
Heads and Tracks

Inside the disc drive itself, there is a Read/Write head, much like a smaller version of that in an ordinary cassette recorder but mounted on the end of an arm. This arm can be moved in and out across the surface of the rotating disc by a stepper motor controlled by the computer, and so can be easily positioned on any required track.

The A: drive on a PCW8256 or 8512 is ***Single Sided, 40-track***. This means that only one of the two surfaces of the disc is used (and so the disc drive can be simpler and cheaper, as it need only have a single Read/Write head), and that the disc is divided up into 40 ***tracks***.

Our 3" discs are rather unusual in the world of computing in that we can turn them over and use the second side in single sided drives! Other types of discs (such as standard 5.25" floppies) cannot be turned over - if they are used in a single- sided drive, their second sides simply cannot be used and are wasted. They are also a lot cheaper than 3" ones, though!

The higher-capacity B: drive, however, is different. This is also the type of drive used as Drive A: on the PCW9512, and is ***Double Sided, 80-track***. This means that ***both*** surfaces of the disc are used at the same time, and so the disc drive needs to have a pair of Read/Write heads, one on each side of the disc. Also, it can pack twice as many tracks onto the same surface area as the less advanced (i.e.cheaper!) 40-track A: drive. Thus a disc in this drive can store four times as much information as a Drive A: disc, and doesn't need to be turned over! In fact, you must ***always*** insert a CF2DD disc the same way round as it was when you formatted it, or the PCW will get confused. It is good practice to ***always*** use these discs with Side A towards the screen - that way you don't have to try and remember "now, which way round did I format this one...?"



The diagram above shows a 40-track CF2 disc. One point to note is that the first track, Track 0, is the ***outermost*** circle on the disc; Track 39 is the ***innermost***. A CF2DD disc looks just the same, but has 80 tracks - twice as many - packed onto the surface. Also, you can imagine a mirror image of the diagram underneath the page, representing the other side of the disc, with, in the case of the double-sided drives, its corresponding Read/Write head in the disc drive.

Getting places

Each track on normal CF2 or CF2DD disc holds exactly 9 ***sectors*** of information, and so the computer can get at any desired sector effectively as fast as to any other simply by calculating which track it is on, stepping the Read/Write head of the disc drive out to the required track and waiting till the sector we want passes under the head, allowing it to be read or written. As the disc rotates at 300 rpm, this doesn't take long! When the PCW is reading or writing data on a double-sided CF2DD disc, it places consecutive tracks on opposite sides of the disc. So, if we consider an 80-track, double-sided disc as having 160 "logical" tracks, numbered from 0 to 159, these are laid out on the disc in order as shown in the table below:

Logical Track Number	Physical Track	Side
0	0	0
1	0	1
2	1	0
3	1	1
4	2	0
5	2	1
6	3	0
.	.	.
.	.	.
.	.	.
157	78	1
158	79	0
159	79	1

When the PCW reports disc errors, it gives the Logical Track number. ToolKit also uses Logical Tracks, so you won't normally have to worry about which side of the disc a particular track or sector is on!

How are files stored on discs ?

Before we can store data usefully on a disc, we need some way of assigning areas of the disc to particular items of data, and also a way of keeping track of where those areas are and what is in them. This ***logical format*** - sitting on top of a disc's physical format of tracks and sectors - introduces the concepts of ***files*** and ***directories***.

On the PCW, we store our data and programs on a disc as ***files***, which all have unique names and so can be recalled easily by us at any time. Each file can be regarded as a high-tech pigeon hole; we can stuff any information we want to keep together (such as all the text in a letter, or all the lines of a program) into a single pigeon hole which we can then find again and look at whenever we want in the future.

To do this, the PCW lumps sectors together into ***blocks*** of 1K or 2K (depending on the type of disc), and allocates storage on the disc to these blocks. It identifies data blocks by number, and from these numbers can calculate the track and sector position of the start of any block.

The PCW stores the block numbers allocated to a particular ***file*** in a ***directory*** which it maintains at the start of the disc. A file is simply a collection of associated data blocks which the computer keeps track of for us - for example, it could be a record from a database or a chapter from your next novel; the PCW neither knows its content nor cares. The directory is simply an indexed list of files stored on the disc and the blocks which comprise them, which allows Locoscript and CP/M to recover files for us and store others.

Block numbers and directory entries are stored and used ***entirely*** internally to the PCW - we never normally need to know about them. However, is important to understand the concept when using ToolKit.

One point to note is that the blocks holding a file do ***not*** need to be contiguous - as the directory holds a record of ***every*** block used by a file, space will be allocated which happens to be available anywhere on the disc.

It may be implied from the above that each file on a disc has a directory entry, and therefore we can store as many files on the disc as we have directory entries available. This is unfortunately not the case! Although their names are listed only once by a "DIR" command, many files have more than one entry in the directory; large files may have several.

Again, we need never normally worry about this, but should be aware that it occurs in order to understand any "Directory Full" error message we may get on a disc which may only have half the theoretical maximum number of files stored on it.

A file's directory entry contains the name we have given the file, information as to whether it has any special attributes such as Read-Only or System, and (most importantly) a list of block numbers which, in order, hold our file. It also contains information such as the Locoscript Group it is in (called a ***User Area*** in CP/M), whether it is in limbo or has been fully deleted.

The detailed structure of directory entries and the way the PCW allocates them to files is complex, and we'll leave a fuller description to Part Four.

Looking after discs

Remember always that the best way of protecting your discs is to ***prevent*** damage occurring rather than trying to cure it after the event - in an ideal world, programs like ToolKit would be redundant! Obviously, some faults occur purely at random and are outwith anyone's control (magnetic corruption can be caused by anything from a power surge to an electrical storm!) but most disasters are avoidable if you follow two simple rules.

Firstly, ***keep frequent backups***. That way, a drama needn't become a crisis! If you lose a disc, you will still have a recent duplicate copy of the information on it, and little or nothing will have been lost. You will also sleep easier at nights and be able to look smug when all around you are tearing their hair out...

If possible, important or irreplacable data should have more than one backup, and these should be kept in different places or even buildings to safeguard against fire, theft or natural disasters. You can buy another computer if it is destroyed or stolen, but replacing your data and software could be a different story!

Secondly, protect your discs from physical harm! Most discs come with a list of "do's and don'ts" on the dust sleeve or box, and these should be followed.

When a disc ***does*** go wrong, it can be for one of two reasons - either the information stored on it has been corrupted by a magnetic field, or it has been physically damaged.

Magnetic Corruption

Data corruption is the easiest to cure - simply reformatting the disc will overwrite it. However, this operation will also overwrite all the uncorrupted data too, so you might have to use ToolKit to recover the rest of the information first.

Corruption can be caused by any stray magnetic field, including the Earth's - most discs will only retain data reliably for a few years, if it is not re-written, because of this. More direct sources of harm are computers themselves (in particular, badly shielded monitors or power supplies - both of which we have in the PCW!), televisions, old style telephones with bells, and hi- fi speakers. This is ***not*** an exhaustive list!

The safest place for discs near a PCW is in the disc drives, as these are shielded! Don't leave discs on top of the monitor or place them near the screen.

If in doubt, keep your discs away from it!

Physical Damage

Physical damage can't be repaired. Often it is visible - inspect discs which have failed for scratches and marks by pushing back the slider on the side edge of the disc case. This will move aside the metal covers over the disc, and you can turn it slowly with two fingers using the plastic centre hub.

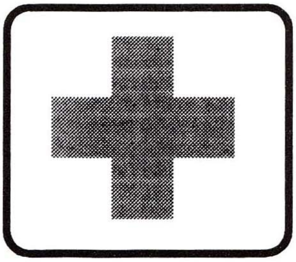
Damage can be caused by dust or dirt inside the disc case scratching the surface or sticking to it, or by oxide on the drive head scoring the disc. Heat, cold and strong sunlight can all warp the disc or damage the magnetic coating, while school chalk and cat or dog hairs are also common hazards which all too many people fail to recognise in time!

If you see complete circles of worn surface on the disc, your heads need cleaned - ***urgently!*** This is done using a special head-cleaning disc containing a pad which you wet with alcohol-based cleaning solution, much like a cassette tape cleaner. Avoid abrasive cleaning discs - they do more harm than good - and ***never*** attempt to scrape adhering particles off a disc surface with your finger or a hard implement! If you ***really*** have to, wet the area lightly with cleaning solution and ***very*** gently attempt to lift the particle with a cotton bud.

Don't ever try and economise by re-using discs suffering from physical damage - throw them away. It's a false economy if it results in unreliability and the loss of programs or data which may be difficult, time-consuming, expensive or even impossible to replace.

PART TWO

First Aid for Discs!



Don't Panic!

REMEMBER - most disc faults ARE recoverable!

- Remove the faulty disc from the computer and push in the Write Protect slider. ***DO NOTHING*** which might result in data being written to the disc until this has been done!

- If you were sensible and kept frequent backups, then you will have lost little or nothing. This is the simplest method of recovery - can you recreate most or all of the lost data easily from elsewhere?

- No? Well, none of us are perfect! Read on.....

First, consider the nature of the problem. There are four basic categories of disaster, and in increasing order of severity these are as follows:

1. Accidental ERAsure of a file.

Here there is nothing wrong with the disc itself, and so long as no data has been written to the disc since you ERAsed the file, recovery is straightforward.

2. Faulty Disc - `"Disc in Drive x is Bad Format"`

This notorious message is often a result of a DISCKIT session which has gone wrong! If you have accidentally copied part of a disc back onto itself by pressing a key too quickly, this error may result and prevent you accessing the disc at all. ToolKit ***can*** recover such a disc, however, relatively painlessly!

3. Damaged or corrupted Data areas.

This is where you were happily loading or saving a file and unexpectedly encounter an error in one of the data sectors holding the file. You will probably lose ***some*** data (that held in the actual sector(s) which are faulty) but should be able to recover the rest of that file and save all the other files on the disc.

4. Damaged or corrupted Directory.

This is the most serious situation, and occurs when a computer fault or program bug has written the wrong information into part or all of the disc's directory, or that particular area of the disc has been corrupted or physically damaged. This will almost certainly require a considerable amount of work to find and recover the data on the disc, but again even in this case it ***is*** possible to salvage most of the data on the disc.

If you get a physical error from the PCW (such as **No Data**, or **Missing Address Mark**, for example), then check to see whether the Track number shown in the error message is **1** (**1** or **2** on a CF2DD or XCF2DD disc); if so, then the faulty sector is likely to be in the directory as this is the area of the disc where it is held. If the Track was further out, then it is a data sector which is faulty (see 2 above) and the problem is not so serious.

Copying a disc with ToolKit

The first step in ***any*** recovery operation is to make a copy of the original, damaged disc. We will then work ***only*** with the copy, ***not*** the original. That way, if we make a mistake we can simply make another copy and start again!

As you will probably have discovered already, none of the normal disc-copying functions on the PCW (such as DISCKIT, XFORMAT or Locoscript 2) will look at a damaged disc! ToolKit ***will*** copy such discs (and will do quite a bit of automatic recovery in the process!)

First, ***write protect*** the faulty disc, if you haven't done so already, so we don't accidentally corrupt it any further!

Boot up CP/M and run ToolKit (see Part Three for details of this if you are not familiar with running CP/M programs).

Insert the faulty disc into Drive A (or Drive B, if you are using a PCW8512 and the disc is CF2DD or XCF2DD format), and log into it with the **L** key (use **D** first to change to Drive B if the disc is there).

If there is any damage to the Configuration Sector, ToolKit will detect this and will prompt for a format to assume. You will then be asked whether you would like ToolKit to fix the Configuration Record. Say **N** (**no**) to this if it happens, as we want to avoid altering the original disc in any way.

Now press the **COPY** key to enter COPY mode, then the **D** key to tell ToolKit to copy the whole disc, and follow any prompts to swap discs if you are copying to a disc in the same drive.

Make sure that the printer is ready and has plenty of paper in it before you start the copying; if ToolKit detects any sectors which are physically damaged or corrupted, it will print a report describing the damage in a Printer Damage Log. Don't worry if you think this printout looks too technical - in general it is only needed when a disc is damaged extensively or in an unusual way. Recovering the most common types of damage with ToolKit is quite straightforward.

When ToolKit has finished the copy, it will automatically write a new Configuration Sector and then log in the new disc, ready to get started!

Recovery Procedures

Un-ERAsing a file

This is the least serious fault ToolKit is likely to be used to repair, as it does not involve corruption to data or damage to the disc.

There are several Un-ERAse utility programs around, mostly in the Public Domain. These are simpler to use than ToolKit - you simply specify the file to be un-ERAsed and the program does it for you - but there are several potentially critical pitfalls in this approach.

Unless the program is quite sophisticated, it is likely to fall over with unpredictable results if it finds that part (but not all) of your ERAsed file has been overwritten with new data. Also, if there is another deleted file with the same name (this isn't as daft or uncommon as it sounds) on the disc, the result will be a nasty mess.

ToolKit requires you to use your eyes and your brain, which we consider preferable to automated disaster...

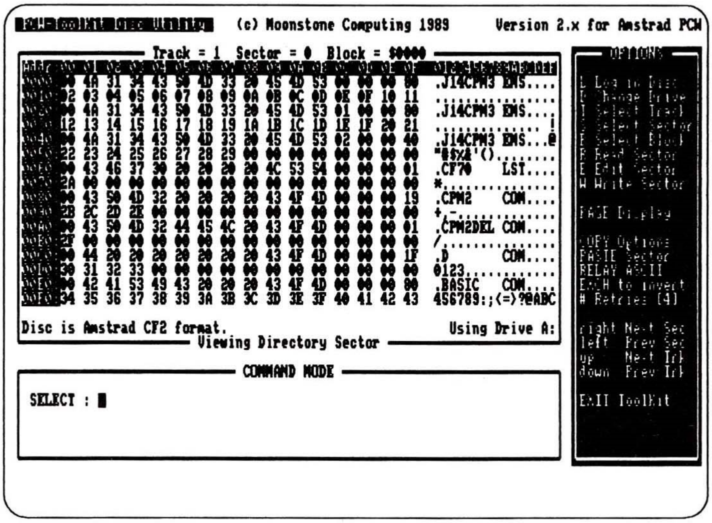
You may find it informative to read the description of Directory Structure given in Section Four; this explains exactly how the PCW ERAses a file. If this is a bit heavy going, though, don't worry; the following straightforward "recipe" will work fine for simple un-ERAsure of a file.

It is important that you have done NOTHING with the disc after you ERAsed the file which might have resulted in data being written to it.

The reason for this is that the PCW doesn't actually delete a file's contents when it is ERAsed; it simply tags the file's directory entries to show that they are now free for re-use. This makes recovery straightforward - we simply change the tag back to normal again - but if any new data has since been written to the disc, is very likely to have used the space freed by the ERAsed file and overwritten some or all it. In such a case, you have virtually no chance of recovering it intact - though you may still be able to find parts of it.

The first step in the recovery process is to copy the disc using ToolKit, as described above. We will then work only with the copy, never with the original

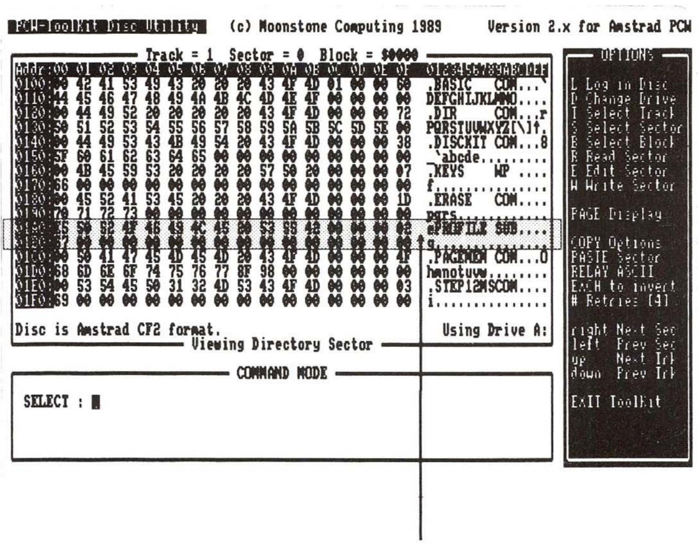
Once this has finished, the screen should look like this :


```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989   Version 2.x for Amstrad PCW

          Track = 1  Sector = 0  Block = $0000
HEX  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 00 4A 31 34 43 50 4D 33 20 45 4D 53 00 00 00 80  .J14CPM3 EMS....
0010 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11  ................
0020 00 4A 31 34 43 50 4D 33 20 45 4D 53 01 00 00 80  .J14CPM3 EMS....
0030 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21  ............... !
0040 00 4A 31 34 43 50 4D 33 20 45 4D 53 02 00 00 40  .J14CPM3 EMS...@
0050 22 23 24 25 26 27 28 29 00 00 00 00 00 00 00 00  "#$%&'()........
0060 00 43 46 37 30 20 20 20 20 4C 53 54 00 00 00 01  .CF70    LST....
0070 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  *...............
0080 00 43 50 4D 32 20 20 20 20 43 4F 4D 00 00 00 19  .CPM2    COM....
0090 2B 2C 2D 2E 00 00 00 00 00 00 00 00 00 00 00 00  +,-.............
00A0 00 43 50 4D 32 44 45 4C 20 43 4F 4D 00 00 00 01  .CPM2DEL COM....
00B0 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  /...............
00C0 00 44 20 20 20 20 20 20 20 43 4F 4D 00 00 00 1F  .D       COM....
00D0 30 31 32 33 00 00 00 00 00 00 00 00 00 00 00 00  0123............
00E0 00 42 41 53 49 43 20 20 20 43 4F 4D 00 00 00 80  .BASIC   COM....
00F0 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43  456789:;<=>?@ABC

Disc is Amstrad CF2 format.                          Using Drive A:
                  Viewing Directory Sector

                       COMMAND MODE
SELECT : █
```

ToolKit has automatically read the first Directory sector; the first half (256 bytes) of this is shown in the Display window.

The first task is to find the first directory entry for the file you ERAsed. Use the ASCII field on the right side of the Display window to find the file's name. If it is not visible, then press **Page** to view the second half of the sector. If it has still not appeared, then you will have to step to the next directory sector and search for it there. Press the **Right Arrow** key to do this, and again check both halves of the new sector for the file's name, using **Page**. Keep repeating this until you have found the first occurrence of the name.

Let's say we have accidentally ERAsed our **PROFILE.SUB** file, and want to recover it. We step through the directory sectors until we find it :



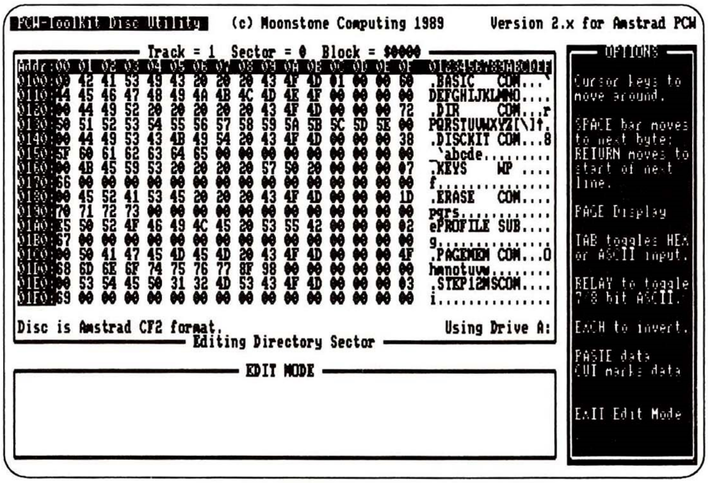
```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989      Version 2.x for Amstrad PCW

 ----------- Track = 1  Sector = 0  Block = $0000 --------------    --- OPTIONS ---
 Addr:00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF    L Log in Disc
 0100:00 42 41 53 49 43 20 20 20 43 4F 4D 01 00 00 60  .BASIC   COM...`   D Change Drive
 0110:44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 00 00 00 00  DEFGHIJKLMNO....   T Select Track
 0120:00 44 49 52 20 20 20 20 20 43 4F 4D 00 00 00 72  .DIR     COM...r   S Select Sector
 0130:50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 00  PQRSTUVWXYZ[\]↑.   B Select Block
 0140:00 44 49 53 43 4B 49 54 20 43 4F 4D 00 00 00 38  .DISCKIT COM...8   R Read Sector
 0150:5F 60 61 62 63 64 65 00 00 00 00 00 00 00 00 00  _`abcde.........   E Edit Sector
 0160:00 4B 45 59 53 20 20 20 20 57 50 20 00 00 00 07  .KEYS    WP ....   W Write Sector
 0170:66 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  f...............
 0180:00 45 52 41 53 45 20 20 20 43 4F 4D 00 00 00 1D  .ERASE   COM....   PAGE Display
 0190:70 71 72 73 00 00 00 00 00 00 00 00 00 00 00 00  pqrs............
 01A0:E5 50 52 4F 46 49 4C 45 20 53 55 42 00 00 00 02  ePROFILE SUB....   COPY Options
 01B0:67 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  g...............   PASTE Sector
 01C0:00 50 41 47 45 4D 45 4D 20 43 4F 4D 00 00 00 4F  .PAGEMEM COM...O   RELAY ASCII
 01D0:68 6D 6E 6F 74 75 76 77 8F 98 00 00 00 00 00 00  hmnotuvw........   EXCH to invert
 01E0:00 53 54 45 50 31 32 4D 53 43 4F 4D 00 00 00 03  .STEP12MSCOM....   # Retries [4]
 01F0:69 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  i...............
                                                                          right Next Sec
 Disc is Amstrad CF2 format.                             Using Drive A:   left  Prev Sec
 ------------------------ Viewing Directory Sector -------------------    up    Next Trk
                                                                          down  Prev Trk
 ---------------------------- COMMAND MODE ---------------------------
                                                                          EXIT ToolKit
 SELECT : █
```

Entry for PROFILE.SUB

You will be able to see that the file is indeed ERAsed, as the first byte of the directory entry (the User Number) is set to **$E5** - the "Deleted File" tag. ($E5 is a Hexadecimal number; if you are not familiar with these, we suggest you read the start of Part Three, where they are explained.)

All we have to do is change this value back to 0 (or to a Locoscript Group number between 0 and 7 for Locoscript files). To do this, press **E** to enter Edit mode.

The screen will now look like this:



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989      Version 2.x for Amstrad PCW

                 Track = 1  Sector = 0  Block = $0000
Addr:00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF     OPTIONS
0100:00 42 41 53 49 43 20 20 20 43 4F 4D 01 00 00 60  .BASIC   COM...`   Cursor keys to
0110:44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 00 00 00 00  DEFGHIJKLMNO....   move around.
0120:00 44 49 52 20 20 20 20 20 43 4F 4D 00 00 00 72  .DIR     COM...r
0130:50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 00  PQRSTUVWXYZ[\]↑.   SPACE bar moves
0140:00 44 49 53 43 4B 49 54 20 43 4F 4D 00 00 00 38  .DISCKIT COM...8   to next byte;
0150:5F 60 61 62 63 64 65 00 00 00 00 00 00 00 00 00  _`abcde.........   RETURN moves to
0160:00 4B 45 59 53 20 20 20 20 57 50 20 00 00 00 07  .KEYS    WP ....   start of next
0170:66 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  f...............   line.
0180:00 45 52 41 53 45 20 20 20 43 4F 4D 00 00 00 1D  .ERASE   COM....
0190:70 71 72 73 00 00 00 00 00 00 00 00 00 00 00 00  pqrs............   PAGE Display
01A0:E5 50 52 4F 46 49 4C 45 20 53 55 42 00 00 00 02  ePROFILE SUB....
01B0:67 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  g...............   TAB toggles HEX
01C0:00 50 41 47 45 4D 45 4D 20 43 4F 4D 00 00 00 4F  .PAGEMEM COM...O   or ASCII input.
01D0:68 6D 6E 6F 74 75 76 77 8F 98 00 00 00 00 00 00  hmnotuvw........
01E0:00 53 54 45 50 31 32 4D 53 43 4F 4D 00 00 00 03  .STEP12MSCOM....   RELAY to toggle
01F0:69 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  i...............   7/8 bit ASCII.

Disc is Amstrad CF2 format.                            Using Drive A:     EXCH to invert.
                   Editing Directory Sector
                                                                          PASTE data
                           EDIT MODE                                      CUT marks data


                                                                          EXIT Edit Mode
```

Use the arrow keys to move the cursor onto the **E** of the **E5** entry, and type **0** twice. You will see the User Number entry change to **00**.

Press **EXIT** to return to Command Mode, then press **W** to write the altered sector back to disc.

One such directory entry is made for every 16K (or part remaining) of the file's size, so if possible calculate how many there should be. You will have to continue to search for each of them and alter them in exactly the same way as described for the first one.

Once you are sure that you have done them all, that's it!

Exit from ToolKit and type

A>**dir** (or **dir b:** if the disc is in Drive "B")

You should now see the file's name present again; and you can proceed to check its contents.

Possible Problems

As was mentioned above, there are some possible difficulties which might make your job a bit harder.

If you have inadvertently written another file to the disc after ERAsing the first one, you will find that some or all of the ERAsed file has been completely overwritten by the new file. In that case, the best you can hope for is that the original file was larger and so one or two of the directory entries for blocks at the end of the file might still be alive.

If you find one, the safest way to recover whatever data is left is to note the Block numbers it holds and use ToolKit's **B** (Select **B**lock) command to go to each of them in turn. You can then move through them, examining them to check that the data is OK, and build them into a file in RAMdisc using the **Paste** function.

Alternatively, you may find that you have too many deleted directory entries for your file! This can occur if there was previously a deleted file with the same name on the disc which had not been completely overwritten. In this case, you will have to check the data each entry refers to and build up a "map" showing which directory entries are actually valid before undeleting them.

Recovering a Bad Format

The second common problem encountered on the PCW is where a disc copying operation has gone wrong, and part of the source disc has been accidentally copied back onto itself. This results in the infamous error message

```
Disc in Drive x is Bad Format
```

The problem in these cases is that during the copying process, the target disc has some of its sectors altered, one of which is the sector holding the Configuration Record on Track 0. This is a touch fatal, as the PCW needs this to tell it information about the disc format!

ToolKit can however fix this without too much difficulty, simply by copying the disc! You should do this now, as described above.

When ToolKit has finished copying the disc, it will automatically write a valid Configuration Record to the copy. This means that the PCW will be able to use the disc again, without you needing to do anything yourself!

Exit from ToolKit and inspect the new disc with CP/M or Locoscript - you should find that all your files are back again.

Possible Problems

Unfortunately, every silver lining has a cloud!

DISCKIT and Locoscript 2 alter another sector on the disc they are copying as well as the Configuration Record. This is at the end of the last part copied in a multi-part copy, which may well be in the middle of one of your files!

You should therefore inspect all text or data files in that region of the disc to check for alteration. Program files on the disc should **not** be run if you haven't found the alteration and corrected it.

The exact location of this altered sector varies depending on how full Drive M was at the time, so there is no way ToolKit can even spot it, never mind fix it automatically - that's where you come in!

The alteration is simple - the data held in the sector is inverted. ToolKit's **EXCH** command allows you to invert (or uninvert!) data in sectors you are examining, so if you feel confident enough you can browse through your files, looking for any obvious inverted pattern. Once you succeed in finding such a sector and have uninverted it on the screen using **EXCH**, simply press **W** to write it back to disc in the new, uninverted form.

That's all there is to it!

Be warned, however, that if the inversion has occurred in the middle of binary data or a program's machine code, then the chances are that you won't be able to tell whether a sector has been inverted or not just by looking at it.

This is what makes programs on such a recovered disc potentially dangerous - if a sector ***has*** been altered and you can't spot it, then simply running that program could do anything - including trashing the disc again!

In general, then, data files are safe to use, as any alteration to them will be discovered as soon as you look at them or try to use them, and is then relatively easy to find with ToolKit and reverse. However, if you have an irreplaceable program on the disc (which shouldn't happen, if you keep proper backups!) and you don't succeed in finding the changed sector on the disc, then leave recovery of this to an expert.

Recovering corrupt or damaged data

The third type of disaster is where one or more sectors containing part of a file's data have become corrupt, or the area of the disc where they are stored has been damaged.

As always, the initial step is to run ToolKit and make a copy of the damaged disc, as described at the start of this section. By default, ToolKit will perform the copy and handle any errors found entirely automatically. The faulty sectors found on the damaged disc will be reported on a Damage Log printout.

During the copying operation, ToolKit will have recovered any damaged data which it could; any unrecoverable sectors will be filled (on the copy disc) with "**?**" characters,

When the copying operation has finished, you can exit ToolKit and begin work on the recovered file. Text (such as Locoscript documents) are obviously the easiest, as you will be able to simply see the pattern of "**?**" characters marking the area of lost data. You can then delete these characters and fill in the missing text.

Possible Problems

Other types of files (database records or spreadsheet files, for example) are harder to deal with and there is little specific guidance we can give here. Each such program varies in the precise way it stores and maintains its data files, and you may have to do some serious hacking with ToolKit's sector editor to glue the file back together again in a form the program will accept. (The Damage Log printout will tell you exactly on the disc where all the damaged sectors are located.)

In such a case, use ToolKit to inspect closely the other data in the file, and try to work out how it is laid out. You may well be able to re-create and insert the lost information, or at least replace it with an acceptable dummy data pattern.

If the damage is in the middle of a program file's machine code, then it won't normally be possible to recover this file and that copy of the program should be ***deleted*** (and certainly ***never*** run).

Rebuilding a damaged Directory

This is the fourth and most serious type of disaster which can befall your discs.

If a program bug or power glitch persuades the PCW to write rubbish over all or part of your directory, then the data on the disc will be inaccessible.

Although the data itself may well not be damaged at all, if the directory is corrupted then the PCW has no way of remembering ***where*** on disc it put the data for each of your files!

As always, the first step is to copy the damaged disc, using ToolKit, as described at the start of this section. If the directory has been physically damaged, rather than just scrambled by a program bug, then the Printer Damage Log will detail this.

Once the original disc has been copied, ToolKit will log to the copy and we can start examining it.

One point to note is that if any of the original disc's directory sectors were physically damaged, ToolKit will have discarded any spurious data read. Instead, those sectors on the copy will have been filled with the value **$E5**, CP/M's uninitialised data pattern which also marks unused directory entries.

Inspect the whole of the Directory area, and determine how much of it has been scrambled or lost, and the exact nature of the damage if it wasn't physical in origin.

"Logical" damage occurs when the all the sectors in the directory were physically read without errors, but on examination either contain garbage or have had their contents subtly altered. This can be the result of a bug in a program, or a power glitch crashing software and fooling it into writing something silly all over your directory.

The basic rebuilding process involves stepping through some or all of the data blocks on the disc, mapping out which files you think they belong to. This can be a long and difficult task, especially if most or all of your directory has been wasted, but is unavoidable.

Beware of red herrings thrown up by old copies of files which were deleted when a more recent version was saved - some of their data may still be on the disc, and life gets rather confusing if you find four copies of the same data!

Once you think you know where all the valid data lives, there are two recovery methods you can use:

1. Reconstruct the Directory

This way, you have to repair or construct directory entries for the files by hand. This is not a trivial task, and is where the knowledge of CP/M directory structure is obviously required.

To rebuild the damaged areas in this way, you will need a fairly good working knowledge of CP/M disc file and directory structure. Section Three covers this ground in reasonable technical detail, and you should also consult any other CP/M reference you have available.

If you have read Part Three and feel confident though, there is absolutely nothing to be lost by having a go - as you should always be working on a ***copy*** of the original disc, nothing will be lost if you get it wrong!

2. Append data sectors to a new file.

The alternative (and simpler, if more time-consuming!) method is to use ToolKit's **Paste** function to build copies of the files from disc, sector by sector, in Drive M.

This function allows you to name new files in RAMdisc to hold the data, and fill each with sectors appended from the damaged disc, one at a time. Once you have this data rebuilt in RAMdisc, you can exit from ToolKit and copy the new files onto a healthy floppy disc.

If you choose to use this method, please read the section describing it in detail in Part Three first!

When all else fails...

If you really can't sort out a disc, then details of a company which offers a professional disc salvage service for Amstrad discs can be obtained from ***Dave's Disk Doctor Service Ltd*** on ***089283-5974***. This company is rather unusual as it is run on the basis that all profits go to charity, principally to the cancer counselling charity **BACUP**.

Other similar (though less charitable!) companies advertise quite widely in the general computer press.

Please note that although we are quite happy to offer general assistance to customers either by phone or by letter (see Introduction), we ***cannot*** offer such recovery services ourselves.

PART THREE

Using ToolKit

Running and stopping the program

PCW-ToolKit runs under the CP/M operating system. This means that you must first load CP/M, as you did when you copied the master disc; ToolKit ***cannot*** be used directly from Locoscript.

PCW 8256/8512 owners ONLY:

Because of bugs in older versions of Amstrad's CP/M BIOS, ToolKit requires that your PCW has BIOS version **1.4** or higher. This has been the standard release version for some time, so most PCW8256/8512s now have this. If however you find you have an older version you will need to contact Amstrad to arrange an update of your System Discs.

Your PCW will display the BIOS version number each time it boots CP/M. If you attempt to run ToolKit under an incompatible BIOS version, this will be detected and the program will terminate safely.

All **PCW9512** machines are compatible with ToolKit.

Once CP/M has loaded and you have the

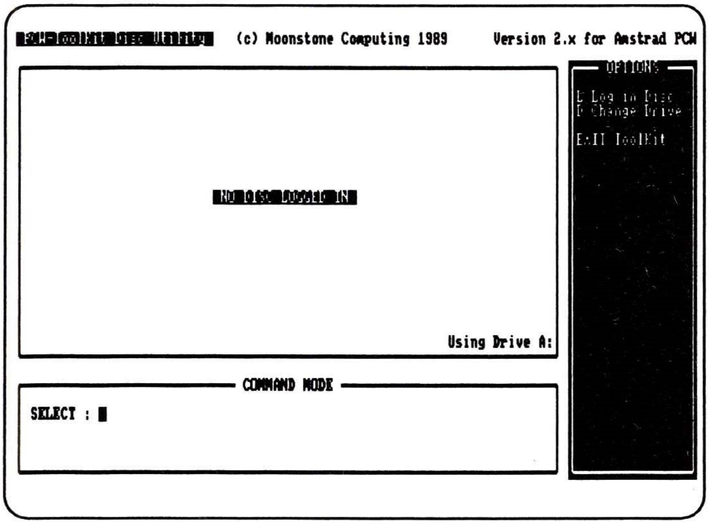
A>

prompt waiting for a command, insert your working copy of the ToolKit disc into Drive A: and type

A>**tk <RETURN>**

This command tells CP/M to load the file containing the ToolKit program (called TK.COM) into memory and execute it.

When ToolKit runs, the screen display will look like this:



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989   Version 2.x for Amstrad PCW

                                                           OPTIONS
                                                           L Log in Disc
                                                           D Change Drive

                                                           EXIT ToolKit

                 NO DISC LOGGED IN

                                        Using Drive A:

                   COMMAND MODE
SELECT : █
```

ToolKit is now waiting for instructions. Note the **COMMAND MODE** message on the box at the bottom of the screen; this is the highest-level menu in ToolKit and is equivalent to the File Manager screen in LocoScript.

From here, you can examine sectors on a disc and move into other modes, such as **EDIT** and **COPY**. In another mode, pressing the **EXIT** key will return you to **COMMAND** mode.

When you are in **COMMAND** mode, pressing the **EXIT** key will stop the program and return you to CP/M. To prevent this happening by mistake, when you press **EXIT** you are asked to confirm that you want to quit the program - type y or **Y** to exit to CP/M, or **n** or **N** if you have second thoughts and want to continue in ToolKit.

Hexadecimal Numbers

The system of numbers we use in everyday life is based around powers and multiples of the number 10, and the ten symbols 0 to 9 are used to construct all our numbers. Thus for example, the number 10 itself actually means "one ten and no units", while 153 means "one hundred, five tens and three units".

This is called the Decimal system, as each digit represents a an additional power of 10 to the digit on its right. (100 is 10 squared, 1,000 is 10 to the power 3, 10,000 is 10 to the power 4 and so on.)

Decimal numbers are easy for us to think in, but are not always so convenient for computers! Because of the organisation of Binary digITs (Bits) used by computers, it is often more useful to represent numbers not in Decimal, but in ***Hexadecimal***.

Instead of being based around powers of 10, Hexadecimal numbers are based around powers of **16**!

This therefore means that we need 16 unique symbols with which to represent such numbers. The convention is that **0** to **9** are used for the first ten, as in Decimal, and the capital letters **A** to **F** are used for the digits equivalent to the Decimal numbers 10 to 15.

In this manual, we'll use the symbol **$** as a prefix to indicate that a number is in ***Hexadecimal***, rather than Decimal, form.

For example, the Decimal number **32** would be written in Hexadecimal (or ***Hex***, for short!) as **$20**. (Remember that this means ("two sixteens and no units"), while the Hex number **$A7** means "**A**, or 10 decimal, sixteens and seven units" - **167** decimal.

Try working out a few conversions for yourself!

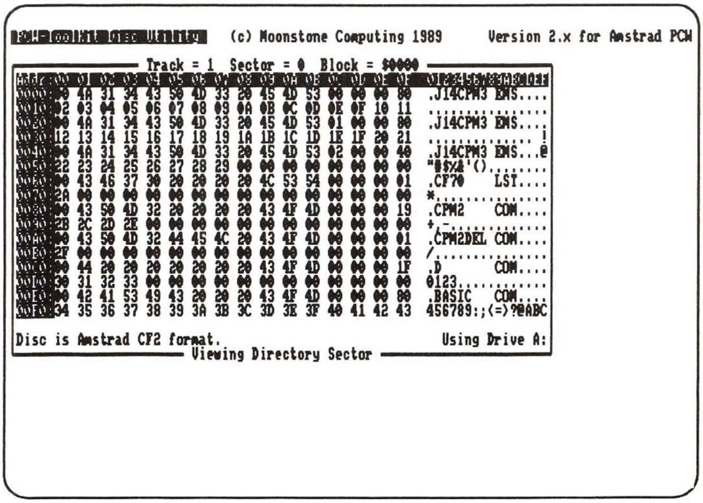
Windows, Modes, Prompts and Commands

ToolKit is designed to be friendly and simple to use. Often the job you are using it to do is complex enough, without the program making life any more difficult!

To achieve this, there is a consistent screen display, with available options always displayed, and a consistent use of keys to avoid confusion - there is nothing worse than a program where the same key performs the opposite functions in two different places!

The Screen

The ToolKit screen display is normally split into three separate ***windows***. The largest is the ***Display*** window, which is used both to display the contents of sectors read from disc and to edit their contents. Once a disc has been logged in and a sector read from it, the Display window looks similar to this:



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989     Version 2.x for Amstrad PCW
 ─────────────── Track = 1  Sector = 0  Block = $0000 ───────────────
 Addr: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
 0000: 00 4A 31 34 43 50 4D 33 20 45 4D 53 00 00 00 80  .J14CPM3 EMS....
 0010: 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11  ................
 0020: 00 4A 31 34 43 50 4D 33 20 45 4D 53 01 00 00 80  .J14CPM3 EMS....
 0030: 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21  ............... !
 0040: 00 4A 31 34 43 50 4D 33 20 45 4D 53 02 00 00 40  .J14CPM3 EMS...@
 0050: 22 23 24 25 26 27 28 29 00 00 00 00 00 00 00 00  "#$%&'()........
 0060: 00 43 46 37 30 20 20 20 20 4C 53 54 00 00 00 01  .CF70    LST....
 0070: 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  *...............
 0080: 00 43 50 4D 32 20 20 20 20 43 4F 4D 00 00 00 19  .CPM2    COM....
 0090: 2B 2C 2D 2E 00 00 00 00 00 00 00 00 00 00 00 00  +,-.............
 00A0: 00 43 50 4D 32 44 45 4C 20 43 4F 4D 00 00 00 01  .CPM2DEL COM....
 00B0: 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  /...............
 00C0: 00 44 20 20 20 20 20 20 20 43 4F 4D 00 00 00 1F  .D       COM....
 00D0: 30 31 32 33 00 00 00 00 00 00 00 00 00 00 00 00  0123............
 00E0: 00 42 41 53 49 43 20 20 20 43 4F 4D 00 00 00 80  .BASIC   COM....
 00F0: 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43  456789:;<=>?@ABC

 Disc is Amstrad CF2 format.                                      Using Drive A:
 ─────────────────────── Viewing Directory Sector ───────────────────────
```

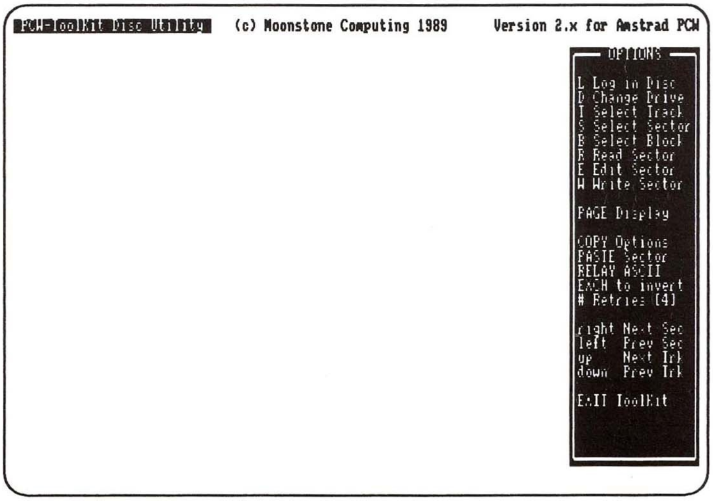
The ***Hex*** field displays each byte held in the sector in Hexadecimal format.

The ***ASCII*** field displays the character whose ASCII code corresponds to the value held in each byte. This makes it much easier to spot and read text files on disc, or examine filenames in the directory.

The Information Line at the top of the window displays the position on disc of the current sector (that is, its Track and Sector numbers), as well as the Block which the sector is part of. The Track and Sector numbers are decimal; the Block number is in Hex.

The Display window only has space to show 256 bytes of a sector's data at a time. As each sector is 512 bytes in size, there are therefore two "pages" which can be viewed. To swap between one and the other, press the **PAGE** key.

The vertical window on the right is the **Options** window, where the options available to you at any time are displayed. In COMMAND mode, after a disc has been logged in, it will look something like this:



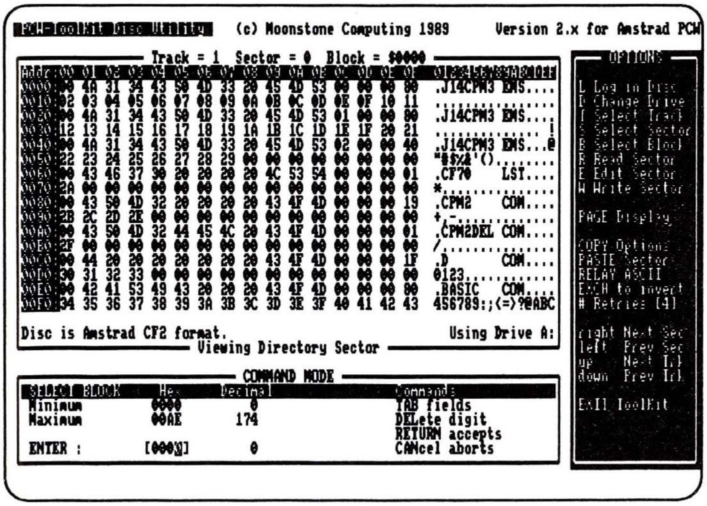
```
PCW-ToolKit Disc Utility    (c) Moonstone Computing 1989    Version 2.x for Amstrad PCW

                                                              OPTIONS
                                                            L Log in Disc
                                                            D Change Drive
                                                            T Select Track
                                                            S Select Sector
                                                            B Select Block
                                                            R Read Sector
                                                            E Edit Sector
                                                            W Write Sector

                                                            PAGE Display

                                                            COPY Options
                                                            PASTE Sector
                                                            RELAY ASCII
                                                            EXCH to invert
                                                            # Retries [4]

                                                            right  Next Sec
                                                            left   Prev Sec
                                                            up     Next Trk
                                                            down   Prev Trk

                                                            EXIT ToolKit
```

The choices **left**, **right**, **up** and **down** all refer to the appropriate arrow keys at the right-hand side of your keyboard.

The exact options displayed in the Options window are different in other modes and at other times; you can always see what options are available to you just by looking here - ToolKit is designed so that you ***don't*** have to refer to the manual constantly!

The bottom part of the screen contains the **Dialogue** window. Here you will see prompts for commands and input, and messages will be displayed here when necessary. Specific options displayed here ***always*** override those shown in the normal Options window, if these are different.

When you are required to enter a numeric value (a track or block number to go to, for example), ToolKit will prompt you for input in the dialogue window with a display something like this :

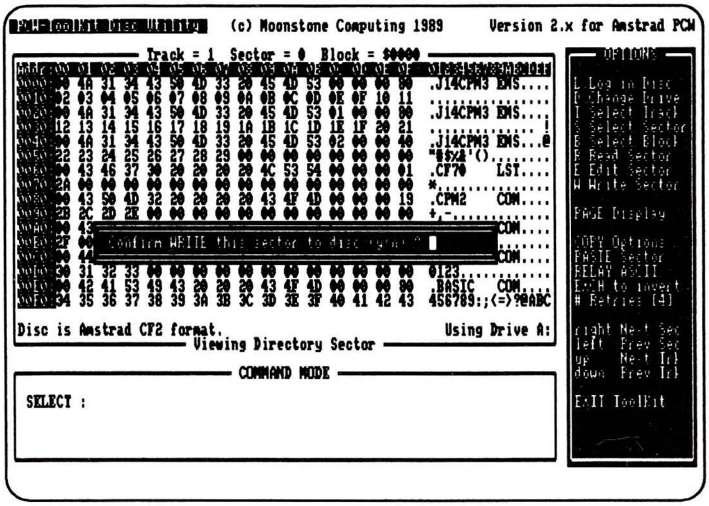


```
PCW-ToolKit Disc Utility  (c) Moonstone Computing 1989      Version 2.x for Amstrad PCW
------------- Track = 1  Sector = 0  Block = $0000 -------------    ---- OPTIONS ----
Addr 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF
0000 00 4A 31 34 43 50 4D 33 20 45 4D 53 00 00 00 80 .J14CPM3 EMS....   L Log in Disc
0010 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 ................   D Change Drive
0020 00 4A 31 34 43 50 4D 33 20 45 4D 53 01 00 00 80 .J14CPM3 EMS....   T Select Track
0030 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 ...............!   S Select Sector
0040 00 4A 31 34 43 50 4D 33 20 45 4D 53 02 00 00 40 .J14CPM3 EMS...@   B Select Block
0050 22 23 24 25 26 27 28 29 00 00 00 00 00 00 00 00 "#$%&'()........   R Read Sector
0060 00 43 46 37 30 20 20 20 20 4C 53 54 00 00 00 01 .CF70    LST....   E Edit Sector
0070 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *...............   W Write Sector
0080 00 43 50 4D 32 20 20 20 20 43 4F 4D 00 00 00 19 .CPM2    COM....
0090 2B 2C 2D 2E 00 00 00 00 00 00 00 00 00 00 00 00 +,-.............   PAGE Display
00A0 00 43 50 4D 32 44 45 4C 20 43 4F 4D 00 00 00 01 .CPM2DEL COM....
00B0 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 /...............   COPY Options
00C0 00 44 20 20 20 20 20 20 20 43 4F 4D 00 00 00 1F .D       COM....   PASTE Sector
00D0 30 31 32 33 00 00 00 00 00 00 00 00 00 00 00 00 0123............   RELAY ASCII
00E0 00 42 41 53 49 43 20 20 20 43 4F 4D 00 00 00 80 .BASIC   COM....   EXCH to invert
00F0 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 456789:;<=>?@ABC   # Retries [4]

Disc is Amstrad CF2 format.                                Using Drive A:   right Next Sec
-------------------- Viewing Directory Sector -------------------           left  Prev Sec
                                                                            up    Next Trk
----------------------------- COMMAND MODE ----------------------           down  Prev Trk
SELECT BLOCK        Hex       Decimal                 Commands
 Minimum           0000           0                   TAB fields            EXIT ToolKit
 Maximum           00AE         174                   DELete digit
                                                      RETURN accepts
 ENTER :          [0000]          0                   CANcel aborts
```

Pressing **TAB** allows you to change between Hex and Decimal entry fields, so you can input numbers in whichever form you are most comfortable using.

Another two windows, the **Confirmation** window and the **Error** window, may appear in the centre of the screen.

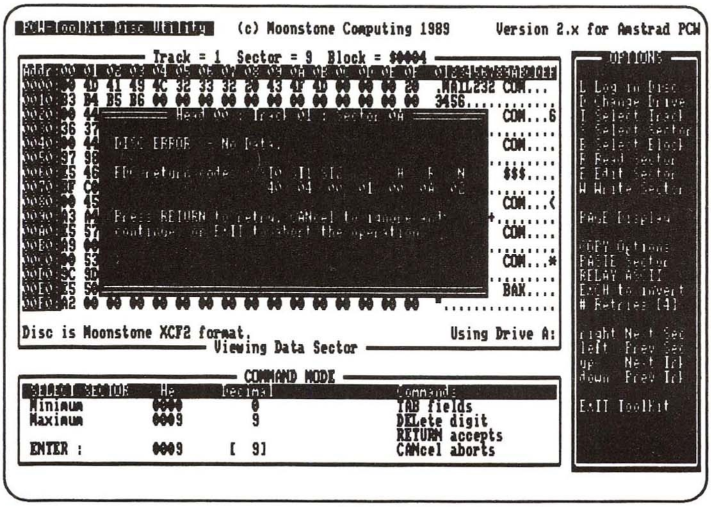
The **Confirmation** window simply asks for confirmation of potentially serious commands (such as writing a sector to disc), and looks like this :



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989     Version 2.x for Amstrad PCW

------------ Track = 1  Sector = 0  Block = $0000 ------------          OPTIONS
Addr 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000 00 4A 31 34 43 50 4D 33 20 45 4D 53 00 00 00 80  .J14CPM3 EMS....   L Log in Disc
0010 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11  ................   D Change Drive
0020 00 4A 31 34 43 50 4D 33 20 45 4D 53 01 00 00 80  .J14CPM3 EMS....   T Select Track
0030 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21  ............... !  S Select Sector
0040 00 4A 31 34 43 50 4D 33 20 45 4D 53 02 00 00 40  .J14CPM3 EMS...@   B Select Block
0050 22 23 24 25 26 27 28 29 00 00 00 00 00 00 00 00  "#$%&'()........   R Read Sector
0060 00 43 46 37 30 20 20 20 20 4C 53 54 00 00 00 01  .CF70    LST....   E Edit Sector
0070 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  *...............   W Write Sector
0080 00 43 50 4D 32 20 20 20 20 43 4F 4D 00 00 00 19  .CPM2    COM....
0090 2B 2C 2D 2E 00 00 00 00 00 00 00 00 00 00 00 00  +,-.............   PAGE Display
00A0 00 43                                                        COM....
00B0 2F 00   Confirm WRITE this sector to disc (y/n) ?                ........   COPY Options
00C0 00 44                                                        COM....   PASTE Sector
00D0 30 31 32 33 00 00 00 00 00 00 00 00 00 00 00 00  0123............   RELAY ASCII
00E0 00 42 41 53 49 43 20 20 20 43 4F 4D 00 00 00 80  .BASIC   COM....   EXCH to invert
00F0 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43  456789:;<=>?@ABC   # Retries [4]

Disc is Amstrad CF2 format.                              Using Drive A:   right Next Sec
-------------------- Viewing Directory Sector --------------------        left  Prev Sec
                                                                          up    Next Trk
------------------------- COMMAND MODE ---------------------------        down  Prev Trk

 SELECT :                                                                 EXIT ToolKit
```

You must press **y** or **Y** to confirm the command, while pressing **n** or **N** will cancel it.

The **Error** window describes problems ToolKit has encountered when reading or writing data, and looks like this :



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989     Version 2.x for Amstrad PCW
           Track = 1  Sector = 9  Block = $0004                          OPTIONS
Addr:00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF   L Log in Disc
0000:00 4D 41 49 4C 32 33 32 20 43 4F 4D 00 00 00 20  .MAIL232 COM...    D Change Drive
0010:33 B4 B5 B6 00 00 00 00 00 00 00 00 00 00 00 00  3456............   T Select Track
0020:00 44 ══ Head 00 : Track 01 : Sector 0A ══════          COM...6     S Select Sector
0030:36 37                                                  .........    B Select Block
0040:00 44  DISC ERROR - No Data.                           COM....      R Read Sector
0050:97 98                                                  .........    E Edit Sector
0060:15 46  FDC return code  ST0 ST1 ST2  C  H  R  N        $$$....      W Write Sector
0070:3F C0                    40  04  00 01 00 0A 02        .........
0080:00 45                                                  COM...<      PAGE Display
0090:A3 A4  Press RETURN to retry, CANcel to ignore and    +.........
00A0:15 57  continue, or EXIT to abort the operation.       COM....      COPY Options
00B0:A9 00                                                  .........    PASTE Sector
00C0:00 53                                                  COM...*      RELAY ASCII
00D0:5C 9D                                                  .........    EXCH to invert
00E0:15 50                                                  BAK....      # Retries (4)
00F0:A2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Disc is Moonstone XCF2 format.                         Using Drive A:    right  Next Sec
               ──── Viewing Data Sector ────                              left   Prev Sec
                                                                         up     Next Trk
                      ──── COMMAND MODE ────                              down   Prev Trk
 SELECT SECTOR      Hex        Decimal             Commands
  Minimum          0000          0                 TAB fields            EXIT ToolKit
  Maximum          0009          9                 DELete digit
                                                   RETURN accepts
  ENTER :          0009       [  9]                CANcel aborts
```

This contains a large amount of useful information about exactly what went wrong, and why. The error type (**Missing Address Mark**, in this example) describes the reason the operation failed; **Part Four** has a full list of possible error types and their likely causes.

For an expert user, the FDC Return Codes (shown in Hex) provide the exact description of the failure, but interpreting these requires a detailed knowledge of the NEC uPD765A Floppy Disc Controller chip used in the PCW, and further explanation of these codes is beyond the scope of this manual.

One important point to note is that the **R** value is the ***physical*** sector number; this will always be one greater than the ***logical*** sector numbers used by ToolKit and CP/M.

Pressing **RETURN** will make ToolKit retry the operation; if it fails again, the error window will again be displayed.

If you press **CAN**cel, then ToolKit will note the fact that the it failed, but will then continue regardless of the error if this is possible. Thus, if you are copying a track which has faulty sectors on it, you can "skip" the bad sectors while copying whenever ToolKit meets them.

Don't worry that CANcelling may be dangerous - if it isn't possible to continue safely, ToolKit will abort and inform you if, for example, data read may be invalid.

If you wish to abort the operation in progress when the error was encountered, press ***EXIT***. This will return you to the screen from which you selected that command.

Modes

Although the basic screen display stays the same throughout ToolKit, the options available to you vary depending on the ***Mode***. The mode you are in is always displayed on an Information Line at the top of the Dialogue window.

The highest-level menu is **COMMAND MODE**, which is active when you enter ToolKit at first. From COMMAND mode, you can select drives and discs, move around a disc examining the contents of tracks, sectors and blocks, copy the contents of sectors into a file in RAMdisc, and select other modes.

ToolKit has two other modes - **EDIT** and **COPY**.

EDIT mode allows you to move the cursor around the Display window, just as in Locoscript, altering any data you wish in the sector you have selected. You will also be able to fill a block with a value, or **CUT** a block for **PASTE**ing into another sector. When you have finished, pressing **EXIT** will return you to COMMAND mode.

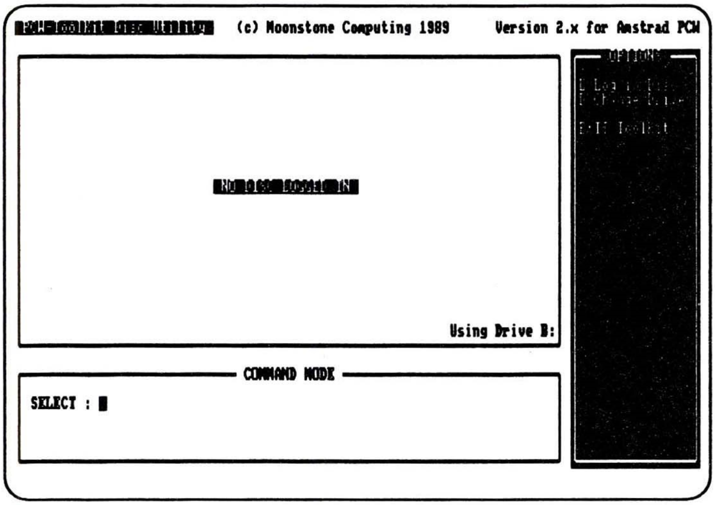
COPY mode allows you to copy either a track or a whole disc, ***whether or not the disc is faulty***. This is one of ToolKit's most useful features, and is crucial to recovering lost or damaged data from such a disc. In this mode, you can also control various system parameters and options.

Selecting a drive

From COMMAND mode, pressing **D** will allow you to use the other physical disc drive (A: or B:). A message at the bottom right of the **Display** window is always present to let you know which drive you are using at any time.

ToolKit is designed to let you examine and change sectors on the two ***physical*** disc drives. It ***isn't*** possible to examine Drive M: with ToolKit.

When you select the other drive using this command, the Display window will clear and the screen will look like this :



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989   Version 2.x for Amstrad PCW

                                                           OPTIONS
                                                          L Log in Disc
                                                          D Change Drive

                                                          E Exit ToolKit

                 NO DISC LOGGED IN




                                    Using Drive B:

                    COMMAND MODE
  SELECT : ▮
```

Note that the **D** option will only be available if your PCW has ***two*** disc drives.

Before you can actually examine, change or copy a disc, you must first ***log it in***. This is the operation by which ToolKit analyses the format of the disc and configures itself automatically for this, and is described in the next section.

Logging in a disc

Before you can use ToolKit to work on a disc, you must log it in by pressing **L**. Until this is done, ToolKit does not know what type of disc it is or what is on it.

When you press **L**, ToolKit examines the system track of the disc in the selected drive to determine its format. If it is a legal Amstrad PCW disc (or a disc created with our XFORMAT utility), then a message detailing this is shown at the bottom left of the Display window.

If the disc is not from a PCW, you will not be able to log it in.

After the disc has been analysed, ToolKit selects the first sector of the directory and read it into memory. The contents of this sector are then displayed.

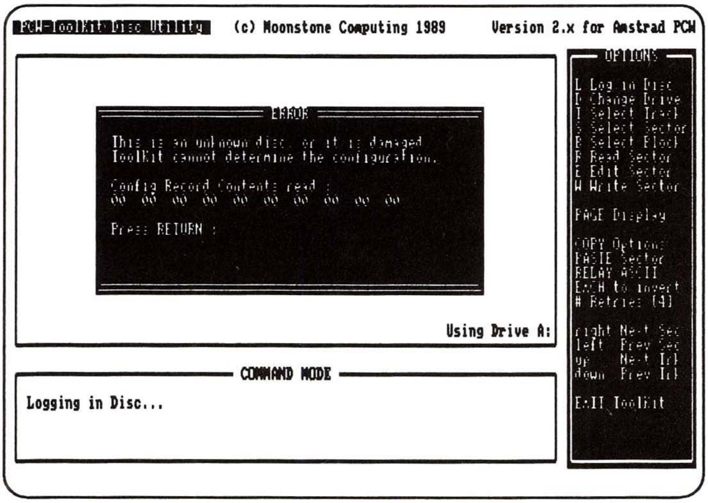
Coping with a damaged system track

If the Configuration Sector on the system track is damaged or corrupt, then ToolKit will be unable to determine the format of the disc. If a disc error occurs while ToolKit is attempting to analyse the disc, then you will be presented with the usual options in the **Error** window of retrying, cancelling or aborting.

If you abort, then the log in operation is aborted and you will be returned to the initial COMMAND mode menu, with no disc logged in.

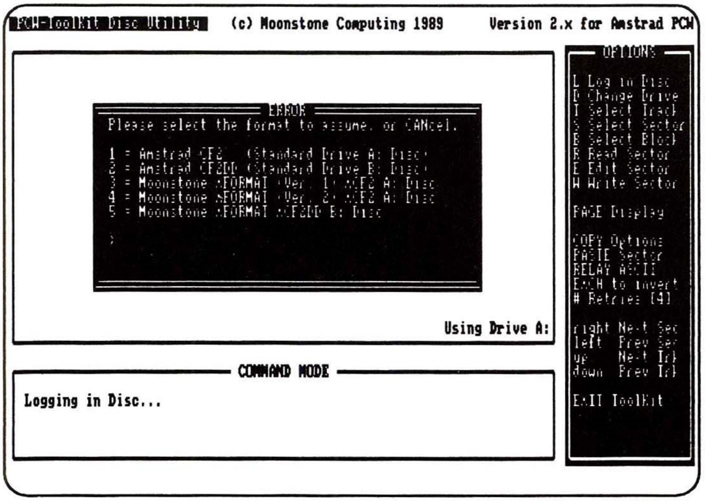
If, however, you CANcel errors and continue, ToolKit will display in the Error window the message shown below..

The list of Hex bytes shown is data read from the ***Configuration Record*** of the disc. This is a data record found at the start of the Boot Sector of every PCW disc, and describes the format of the disc. The exact contents of this record are specified fully in Part Four. Note that after most physical disc errors, the values of these bytes shown will probably be invalid.



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989   Version 2.x for Amstrad PCW

                        ERROR                                         OPTIONS
  This is an unknown disc, or it is damaged.                          L Log in Disc
  ToolKit cannot determine the configuration.                         D Change Drive
                                                                      T Select Track
  Config Record Contents read :                                       S Select Sector
  00  00  00  00  00  00  00  00  00  00                              B Select Block
                                                                      R Read Sector
  Press RETURN :                                                      E Edit Sector
                                                                      W Write Sector

                                                                      PAGE Display

                                                                      COPY Options
                                                                      PASTE Sector
                                                                      RELAY ASCII
                                                                      EXCH to invert
                                                                      # Retries [4]
                                                 Using Drive A:
                                                                      right Next Sec
                       COMMAND MODE                                   left  Prev Sec
Logging in Disc...                                                    up    Next Trk
                                                                      down  Prev Trk

                                                                      EXIT ToolKit
```

This error window is also displayed if ToolKit was able to read the disc without any physical errors occurring, but detected an illegal configuration record. In this case, the bytes shown ***will*** be those read from disc.

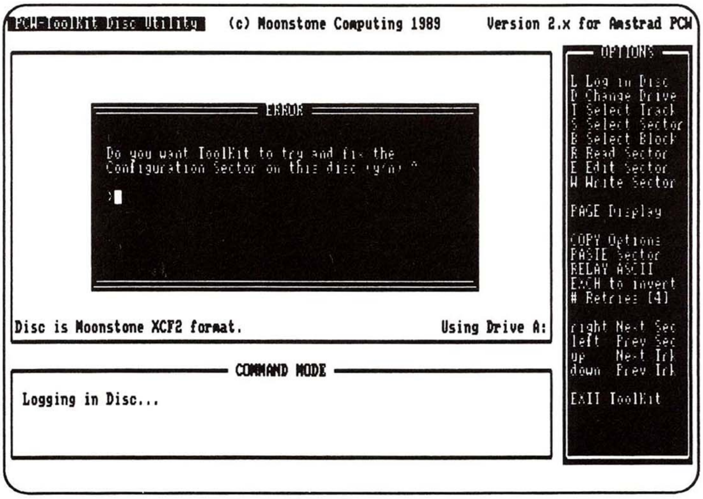


```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989   Version 2.x for Amstrad PCW

                        ERROR                                         OPTIONS
  Please select the format to assume, or CANcel.                      L Log in Disc
                                                                      D Change Drive
  1 = Amstrad CF2   (Standard Drive A: Disc)                          T Select Track
  2 = Amstrad CF2DD (Standard Drive B: Disc)                          S Select Sector
  3 = Moonstone XFORMAT (Ver. 1) XCF2 A: Disc                         B Select Block
  4 = Moonstone XFORMAT (Ver. 2) XCF2 A: Disc                         R Read Sector
  5 = Moonstone XFORMAT XCF2DD B: Disc                                E Edit Sector
                                                                      W Write Sector
  >
                                                                      PAGE Display

                                                                      COPY Options
                                                                      PASTE Sector
                                                                      RELAY ASCII
                                                                      EXCH to invert
                                                                      # Retries [4]
                                                 Using Drive A:
                                                                      right Next Sec
                       COMMAND MODE                                   left  Prev Sec
Logging in Disc...                                                    up    Next Trk
                                                                      down  Prev Trk

                                                                      EXIT ToolKit
```

Whatever the cause of the problem, ToolKit cannot work out the format on its own. Consequently, after you press **RETURN** the above message is displayed.

If you now press **CAN**cel, then the login operation will be aborted. Otherwise, you should type the number corresponding to the type of disc you are trying to use. Note that the descriptions of the formats refer to the different drives on the PCW8512; on a PCW9512, the discs normally used in the A: drive are CF2DD format.

If you select a format for ToolKit to assume correct for the disc, a third message will be displayed :



```
PCW-ToolKit Disc Utility    (c) Moonstone Computing 1989    Version 2.x for Amstrad PCW

                                ERROR
     Do you want ToolKit to try and fix the
     Configuration Sector on this disc (y/n) ?

     >

Disc is Moonstone XCF2 format.                         Using Drive A:

                          COMMAND MODE
 Logging in Disc...
```

```
OPTIONS
L Log in Disc
D Change Drive
T Select Track
S Select Sector
B Select Block
R Read Sector
E Edit Sector
W Write Sector

PAGE Display

COPY Options
PASTE Sector
RELAY ASCII
EXCH to invert
# Retries [4]

right Next Sec
left  Prev Sec
up    Next Trk
down  Prev Trk

EXIT ToolKit
```

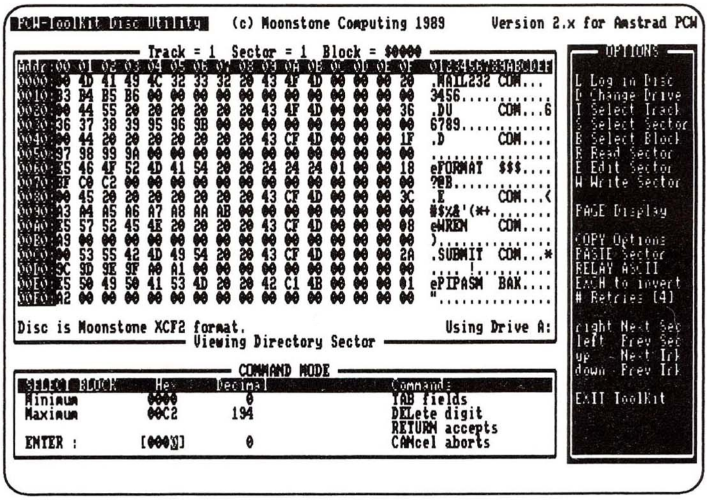
Here, ToolKit is offering to write a Configuration Record to the disc which will be valid for the format you have selected. If the original damage to the configuration sector was serious, ToolKit will automatically reformat the system track before writing the new copy of the sector.

Once you have selected whether or not to attempt to fix the Configuration Record, ToolKit will attempt to read in the first sector of the directory, as normal.If this area of the disc is also faulty, then further physical errors will occur, and again you should consult Part Two for suggested ways of repairing the damage.

Selecting a Block

If you want to examine or modify the contents of a particular block from a file, you can ask ToolKit to "goto" it directly by using the **B Select Block** command from **COMMAND** mode.

When you select this command, ToolKit will prompt you to enter the number of the block you want to go to, like this :



```
PCW-ToolKit Disc Utility    (c) Moonstone Computing 1989      Version 2.x for Amstrad PCW

 ------------ Track = 1  Sector = 1  Block = $0000 ------------    --- OPTIONS ---
 Addr 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
 0000:00 4D 41 49 4C 32 33 32 20 43 4F 4D 00 00 00 20  .MAIL232 COM...     L Log in Disc
 0010:B3 B4 B5 B6 00 00 00 00 00 00 00 00 00 00 00 00  3456............    D Change Drive
 0020:00 44 55 20 20 20 20 20 20 43 4F 4D 00 00 00 36  .DU      COM...6    T Select Track
 0030:36 37 38 39 95 96 9B 00 00 00 00 00 00 00 00 00  6789............    S Select Sector
 0040:00 44 20 20 20 20 20 20 20 43 CF 4D 00 00 00 1F  .D       COM....    B Select Block
 0050:97 98 99 9A 00 00 00 00 00 00 00 00 00 00 00 00  ................    R Read Sector
 0060:E5 46 4F 52 4D 41 54 20 20 24 24 24 01 00 00 18  eFORMAT  $$$....    E Edit Sector
 0070:BF C0 C2 00 00 00 00 00 00 00 00 00 00 00 00 00  ?@B.............    W Write Sector
 0080:00 45 20 20 20 20 20 20 20 43 CF 4D 00 00 00 3C  .E       COM...<
 0090:A3 A4 A5 A6 A7 A8 AA AB 00 00 00 00 00 00 00 00  #$%&'(*+........    PAGE Display
 00A0:E5 57 52 45 4E 20 20 20 20 43 CF 4D 00 00 00 08  eWREN    COM....
 00B0:A9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  )...............    COPY Options
 00C0:00 53 55 42 4D 49 54 20 20 43 CF 4D 00 00 00 2A  .SUBMIT  COM...*    PASTE Sector
 00D0:9C 9D 9E 9F A0 A1 00 00 00 00 00 00 00 00 00 00  .... !..........    RELAY ASCII
 00E0:E5 50 49 50 41 53 4D 20 20 42 C1 4B 00 00 00 01  ePIPASM  BAK....    EACH to invert
 00F0:A2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  "...............    # Retries [4]

 Disc is Moonstone XCF2 format.                        Using Drive A:       right Next Sec
 --------------------- Viewing Directory Sector ----------------------     left  Prev Sec
                                                                            up    Next Trk
 ------------------------------ COMMAND MODE ------------------------       down  Prev Trk
 SELECT BLOCK          Hex       Decimal                Commands
   Minimum             0000         0                   TAB fields          EXIT ToolKit
   Maximum             00C2       194                   DELete digit
                                                        RETURN accepts
   ENTER :            [0000]        0                   CANcel aborts
```

The default mode of entry here is ***Hexadecimal***, as normally you will be using the hexadecimal block numbers as they are stored and displayed in the directory. However, if you want to enter a block number in Decimal, press TAB to toggle between the two input fields.

The **DEL** key will work, allowing you to correct mistakes; when you have entered the correct block number, press **RETURN** and ToolKit will move the heads to that block and display its first sector.

If you change your mind about the whole thing, the **CAN** key cancels the command and returns you to the **COMMAND** mode.

Selecting a Track

It is also possible to select a particular Physical Track on the disc for attention, rather than a block of a file. This is useful if, for example, we have a disc which has a fault on it which we want to investigate.

This is the sort of problem that arises out of the blue when you suddenly get a CP/M BIOS error message such as

```
A: track 12, sector 3 no data - Retry, Ignore or Cancel?
```

in the middle of a program, or Locoscript aborts in the middle of loading a file with a similar message.

The **T Select Track** command lets us go directly to the offending track and investigate. If we press **T**, we will again be prompted to enter the number of the track we want to go to, either in Hex or Decimal. In this case, the default mode is ***Decimal***, as this is how track numbers are displayed in error messages and by ToolKit.

So, with the above example, if we type **12**, then ToolKit will move the heads to logical track 12 and attempt to read the first sector on the track and display its contents.

If one sector on a track is faulty, it is quite possible that others may be as well, so this operation may result in the Error window being displayed as described earlier. CANcelling any error will leave us at the offending track, but with no sector read. We can then use the left and right arrow keys to move to adjacent sectors on the track, or use the **S Select Sector** command to explicitly select a sector and attempt to read it.

Selecting a Sector

The **S Select Sector** command allows us to go to a particular sector within the track we are on. As before, we are prompted to enter the sector number in either Hex or Decimal. The default in this case is ***decimal***, as this is how sector numbers are reported in error messages.

Like CP/M and Locoscript, ToolKit uses ***logical*** sector numbers, where the first sector on each track is numbered zero.

Once you have entered a legal sector number, ToolKit will attempt to read that sector from disc.

You may find that it is often easier simply to press the left or right cursor keys to move from sector to sector than explicitly entering a sector number; the result is the same.

If a sector has a fault, then ToolKit will probably fail to read it. However, don't give up straight away - Retry several times, remove the disc and give it a sharp knock on the top of the desk to dislodge any dust which may be adhering to the surface inside and try again. Also, press the slider on the side edge of the disc back to uncover the surface and inspect it for visible damage, turning it slowly by the centre hub. ***Don't*** ever touch the disc surface itself.

If a sector is completely unreadable, then check all the other sectors on the same track, and the adjacent tracks on either side of it. This will show whether the problem is more widespread than a single sector (and therefore probably due to physical damage of some kind, even if it isn't visible) or is restricted to the one sector (and so is most likely just magnetic corruption of data).

The latter can be cured by reformatting the disc (***after*** you have used ToolKit to recover the rest of the data on it!), but physical damage means that the disc is ruined, and after recovering the all data possible from it should be thrown away.

Reading a sector from disc

The **R Read Sector** command will read the current sector from disc. Normally, this is done automatically whenever a new sector, track or block is selected and so this command rarely needs to be used explicitly.

Where this facility is useful is if you have edited the contents of a sector, and have made a mistake. Often, rather than re- edit, it is easier to simply re-read the sector from disc using this command and start again. Obviously, if you had written the sector to disc after the changes had been made, then this would not work since the original sector would have been overwritten.

Writing a sector to disc

The **W Write Sector** command will write the current sector from memory to disc. The original contents of the sector on disc will be overwritten with any changes made in the editor.

ToolKit always requests confirmation of a write command before proceeding, with the message

```
Confirm WRITE this sector to disc (y/n) :
```

If you reply **n** or **N**, the operation will be aborted, while y or **Y** will tell ToolKit to proceed and write the sector.

If you have used **EDIT** mode to alter the contents of a sector, you ***must*** use this command if you want to write the modified data back to disc; this is ***not*** done automatically.

Selecting the alternate ASCII field display

In both **COMMAND** and **EDIT** modes, the **RELAY** key allows you to toggle the way the ASCII field of the Display Window is shown.

By default, only bytes whose values correspond to characters in the standard ASCII set are displayed in the ASCII field. These values lie in the range **32 ($20)** to **127 ($7F)** inclusive. Any byte whose value is outwith these limits has its ASCII equivalent shown simply as a dot (**.**).

However, pressing the RELAY key will change this. ***All*** byte values will then be shown as their character equivalents from the PCW's extended character set. This has 256 characters (in other words, all possible values are associated with a character), and includes the standard ASCII range as a subset. The extra characters are made up of foreign language symbols and graphics characters.

RELAY is a toggle, so pressing it again will revert to displaying standard ASCII only.

Inverting Data

In both **COMMAND** and **EDIT** modes, the **EXCH** key toggles the way data is interpreted. The initial setting is to do nothing, but if EXCH is pressed once, then all data read will be ***inverted***.before being displayed. Inversion is a logical one's- complement; every 0 bit is made into a 1 and every 1 is made into a 0.

EXCH is a toggle; Pressing it a second time returns to "normal".

This facility is useful when recovering a disc which has accidentally been copied onto itself; this is discussed more fully in Section Two.

Altering the Disc Error Retry count

By default, when ToolKit meets a faulty sector it will make four retry attempts to complete the operation successfully. Only if all of these retries fail will an error be recorded and a message displayed if appropriate.

You can alter the number of retry attempts ToolKit will make on each faulty sector by pressing the # key and entering a value between 0 and 9.

Editing a sector

Pressing the **E Edit** key from **COMMAND** mode switches ToolKit into **EDIT MODE**. A completely different set of options is then presented in the Options Window, and the cursor is placed over the first byte of the sector in the Display Window.

Within EDIT mode, you can move around the sector, altering any of the contents. You can also mark a block and either fill it with a value or copy it into a buffer. Alternatively, you could overwrite part or all of the sector with data already held in the buffer, perhaps from a different sector.

To exit from **EDIT** mode and return to **COMMAND** mode, press **EXIT**.

Moving About

The cursor keys move the cursor around the sector in all directions. Pressing **RETURN** will move the cursor onto the first byte on the next line, while pressing **SPACE** moves to the start of the next byte.

If the cursor is on the bottom line of the display and you move it down, it will ***wrap round*** and go to the corresponding position on the top line. Similarly, if you move the cursor up while on the top line of the display it will wrap round to the bottom line.

As in COMMAND mode, the **Page** command will display the other part of the sector for editing.

Changing Fields

ToolKit lets you edit in either the Hexadecimal display field **or** the ASCII field. If you want to modify or insert text, then editing the ASCII field makes this very straightforward, while if you are modifying numerical data then Hex input is more suitable.

You can toggle between the two fields at will by pressing the **TAB** key.

While in the Hex field, only Hex digits (numbers in the range **0** to **9**, and letters in the range **A** to **F**) will be accepted.

In the ASCII field, any ASCII character with a code between **$20** (32 decimal) and **$7F** (127 decimal) will be accepted. This range includes all the standard characters found on the keyboard, but does **not** in general allow any of the PCW's special characters to be input as ASCII (these mostly have codes of **$80** or higher).

Altering Data

You may freely alter any of the data stored in the sector by simply overwriting it with values typed on the keyboard. The keys accepted vary depending whether you are editing Hex or ASCII; this was detailed above.

Note that Hex field is exactly that - numbers are displayed ***and edited*** as ***Hexadecimal***, not ordinary decimal. If you want to enter a number which you know only in its decimal form, you must first convert it to its Hex equivalent. For example, the decimal number **32** has a hex value of **20**, and must be entered as the latter.

Hexadecimal numbers are explained in detail at the start of Part Three, and a Decimal to Hex conversion table is included at the back of this manual as an aid to those not used to thinking like a computer!

One last point to note is that **SPACE** is a valid ASCII character, and will be accepted as such if you are editing the ASCII field. Thus you should **not** use the SPACE bar just to move to the next byte, as you can when working in the Hex field - instead, you should use the Right Arrow key.

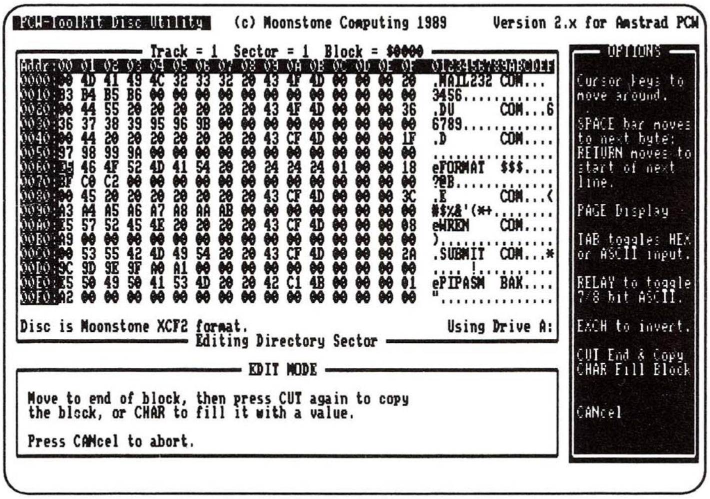
Filling a Block

ToolKit's editor allows you to mark a block of data within a sector and either fill it with a constant value or copy it into a buffer for insertion elsewhere later.

This copying function is detailed in the next section, **Copying Data**.

To fill a block with a value (to initialise a sector to contain all 00s, for example) we must first move the cursor to the first byte of the block and tell ToolKit that we want to make this position the start of the block.

We do this by using the cursor keys to move the cursor to the chosen position, and pressing the **CUT** key. The Options Window will change slightly, and the following message will be displayed in the Dialogue Window at the bottom of the screen :

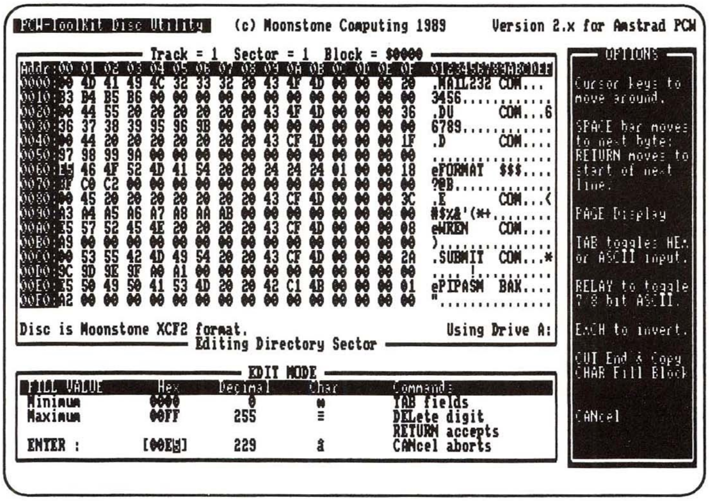


```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989     Version 2.x for Amstrad PCW

 ──────── Track = 1  Sector = 1  Block = $0000 ─────────          ── OPTIONS ──
Addr: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000: 00 4D 41 49 4C 32 33 32 20 43 4F 4D 00 00 00 20  .MAIL232 COM...      Cursor keys to
0010: B3 B4 B5 B6 00 00 00 00 00 00 00 00 00 00 00 00  3456............     move around.
0020: 00 44 55 20 20 20 20 20 20 43 4F 4D 00 00 00 36  .DU      COM...6
0030: 36 37 38 39 95 96 9B 00 00 00 00 00 00 00 00 00  6789............     SPACE bar moves
0040: 00 44 20 20 20 20 20 20 20 43 CF 4D 00 00 00 1F  .D       COM....     to next byte;
0050: 97 98 99 9A 00 00 00 00 00 00 00 00 00 00 00 00  ................     RETURN moves to
0060: E5 46 4F 52 4D 41 54 20 20 24 24 24 01 00 00 18  eFORMAT  $$$....     start of next
0070: BF C0 C2 00 00 00 00 00 00 00 00 00 00 00 00 00  ?@B.............     line.
0080: 00 45 20 20 20 20 20 20 20 43 CF 4D 00 00 00 3C  .E       COM...<
0090: A3 A4 A5 A6 A7 A8 AA AB 00 00 00 00 00 00 00 00  #$%&'(*+........     PAGE Display
00A0: E5 57 52 45 4E 20 20 20 20 43 CF 4D 00 00 00 08  eWREN    COM....
00B0: A9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  )...............     TAB toggles HEX
00C0: 00 53 55 42 4D 49 54 20 20 43 CF 4D 00 00 00 2A  .SUBMIT  COM...*     or ASCII input.
00D0: 9C 9D 9E 9F A0 A1 00 00 00 00 00 00 00 00 00 00  .... !..........
00E0: E5 50 49 50 41 53 4D 20 20 42 C1 4B 00 00 00 01  ePIPASM  BAK....     RELAY to toggle
00F0: A2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  "...............     7/8 bit ASCII.

Disc is Moonstone XCF2 format.                         Using Drive A:      EXCH to invert.
 ─────────────────── Editing Directory Sector ───────────────────
                                                                          CUT End & Copy
 ───────────────────────── EDIT MODE ────────────────────────────          CHAR Fill Block

  Move to end of block, then press CUT again to copy
  the block, or CHAR to fill it with a value.                             CANcel

  Press CANcel to abort.
```

You should now use the cursor keys (and **Page**, if necessary) to move to the last byte in the block you want to mark. You can also use **TAB** and move around in either the Hex or the ASCII field.

When you have positioned the cursor over the last byte in the block, press **CHAR**.

ToolKit will then prompt for the value or ASCII character with which to fill the block :



```
PCW-ToolKit Disc Utility    (c) Moonstone Computing 1989      Version 2.x for Amstrad PCW

        Track = 1  Sector = 1  Block = $0000
Addr:00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000:00 4D 41 49 4C 32 33 32 20 43 4F 4D 00 00 00 20  .MAIL232 COM...
0010:B3 B4 B5 B6 00 00 00 00 00 00 00 00 00 00 00 00  3456............
0020:00 44 55 20 20 20 20 20 20 43 4F 4D 00 00 00 36  .DU      COM...6
0030:36 37 38 39 95 96 9B 00 00 00 00 00 00 00 00 00  6789............
0040:00 44 20 20 20 20 20 20 20 43 CF 4D 00 00 00 1F  .D       COM....
0050:97 98 99 9A 00 00 00 00 00 00 00 00 00 00 00 00  ................
0060:E5 46 4F 52 4D 41 54 20 20 24 24 24 01 00 00 18  eFORMAT  $$$....
0070:BF C0 C2 00 00 00 00 00 00 00 00 00 00 00 00 00  ?@B.............
0080:00 45 20 20 20 20 20 20 20 43 CF 4D 00 00 00 3C  .E       COM...<
0090:A3 A4 A5 A6 A7 A8 AA AB 00 00 00 00 00 00 00 00  #$%&'(*+........
00A0:E5 57 52 45 4E 20 20 20 20 43 CF 4D 00 00 00 08  eWREN    COM....
00B0:A9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  )...............
00C0:00 53 55 42 4D 49 54 20 20 43 CF 4D 00 00 00 2A  .SUBMIT  COM...*
00D0:9C 9D 9E 9F A0 A1 00 00 00 00 00 00 00 00 00 00  .... !..........
00E0:E5 50 49 50 41 53 4D 20 20 42 C1 4B 00 00 00 01  ePIPASM  BAK....
00F0:A2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  "...............

Disc is Moonstone XCF2 format.                                Using Drive A:
                    Editing Directory Sector

                            EDIT MODE
FILL VALUE         Hex      Decimal      Char          Commands
 Minimum          0000         0          @            TAB fields
 Maximum          00FF       255          ≡            DELete digit
                                                       RETURN accepts
 ENTER :         [00E5]      229          å            CANcel aborts
```

```
OPTIONS

Cursor keys to
move around.

SPACE bar moves
to next byte;
RETURN moves to
start of next
line.

PAGE Display

TAB toggles HEX
or ASCII input.

RELAY to toggle
7/8 bit ASCII.

EXCH to invert.

CUT End & Copy
CHAR Fill Block

CANcel
```

The default mode of input depends on the Display field you were editing when you pressed **CUT** and will therefore either be Hex or ASCII. Pressing **TAB**, however, will move the cursor to the next input field, allowing you to input the fill value as Hex, Decimal or an ASCII character typed on the keyboard with a value between 32 and 127 (decimal).

The **DEL**ete key will work, allowing you to correct mistakes; press **RETURN** when you have entered the correct value or character and ToolKit will fill the block you have marked with that value or character.

Pressing **CAN** will cancel the command and return you to normal editing mode.

Copying Data

Toolkit provides a variety of functions to help you copy and recover data from discs in amounts varying from a few bytes through to a whole disc, and fall into four main groups.

Using CUT and PASTE in the Editor

The first method of copying data from all or part of a sector is available while you are in **EDIT** mode.

We use the **CUT** function as before, when we saw how to fill a block within a sector with a constant value or character. You must first position the cursor over the first byte you want to be contained in the block, and press the **CUT** key.

The Options window will then show the two choices now available, and the Dialogue window will tell us to move to the end of the block and select one of the options.

Before, we selected the **CHAR** function to Fill the block with a value. Now, press the **CUT** key **again** and the block will be copied into ToolKit's buffer. Note that this does not alter the original block in any way.

However, you can now exit **EDIT** mode and go to any other sector on the disc (or on a different disc) and copy the contents of the block to anywhere in that sector.

To do this, you must have loaded the desired sector and have re-entered **EDIT** mode. Move the cursor to the start of the position where you want to insert the data. Now, if you press the **PASTE** key, the data you cut out and placed in the buffer will then be inserted, starting at the position of the cursor, overwriting whatever was there previously.

Using PASTE to append sectors to a file.

The second, and simpler, method of copying data can be carried out directly from **COMMAND** mode by pressing a single key - **PASTE**.

This will append the entire sector currently loaded to the end of a file in Drive M:. If the file doesn't yet exist (as it won't for the first sector "rolled out" in this way, it is automatically created and the first sector put into it.

This allows you to build up a file in RAMdisc containing data from sectors anywhere on a disc, and can be most useful in rebuilding a file whose directory entry has been corrupted. If you can find the contents of a file on disc, you can roll it out sector-by-sector into a new file in RAMdisc. This file can then be copied out onto a physical disc again.

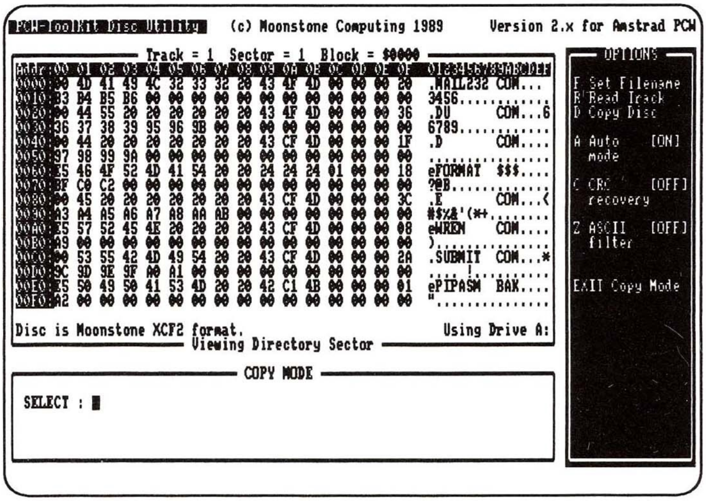
Note that this is the one instance in ToolKit where a key - **PASTE** - has a duplicate function. In **COMMAND** mode, **PASTE** will append a sector to a RAMdisc file, while in **EDIT** mode, **PASTE** will insert any data held in the Cut & Paste buffer as described above.

By default, the file used to contain the data is called **M:IMAGE.DSK**, but the name of the file currently used by this function can be set and changed easily by the **F Set Filename** function in COPY mode - this is described below.

COPY Options

The other two methods ToolKit provides for copying data - by whole Track and by whole Disc - are accessed from the **COPY** mode menu.

Pressing the **COPY** key from **COMMAND** mode will put ToolKit into **COPY** mode, and the screen will look like this :



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989      Version 2.x for Amstrad PCW

 ------------ Track = 1  Sector = 1  Block = $0000 ------------   ---- OPTIONS ----
 Addr:00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
 0000:00 4D 41 49 4C 32 33 32 20 43 4F 4D 00 00 00 20  .MAIL232 COM...    F Set Filename
 0010:B3 B4 B5 B6 00 00 00 00 00 00 00 00 00 00 00 00  3456............   R Read Track
 0020:00 44 55 20 20 20 20 20 20 43 4F 4D 00 00 00 36  .DU      COM...6   D Copy Disc
 0030:36 37 38 39 95 96 9B 00 00 00 00 00 00 00 00 00  6789............
 0040:00 44 20 20 20 20 20 20 20 43 CF 4D 00 00 00 1F  .D       COM....   A Auto     [ON]
 0050:97 98 99 9A 00 00 00 00 00 00 00 00 00 00 00 00  ................     mode
 0060:E5 46 4F 52 4D 41 54 20 20 24 24 24 01 00 00 18  eFORMAT  $$$....
 0070:BF C0 C2 00 00 00 00 00 00 00 00 00 00 00 00 00  ?@B.............   C CRC      [OFF]
 0080:00 45 20 20 20 20 20 20 20 43 CF 4D 00 00 00 3C  .E       COM...<     recovery
 0090:A3 A4 A5 A6 A7 A8 AA AB 00 00 00 00 00 00 00 00  #$%&'(*+........
 00A0:E5 57 52 45 4E 20 20 20 20 43 CF 4D 00 00 00 08  eWREN    COM....   Z ASCII    [OFF]
 00B0:A9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  )...............     filter
 00C0:00 53 55 42 4D 49 54 20 20 43 CF 4D 00 00 00 2A  .SUBMIT  COM...*
 00D0:9C 9D 9E 9F A0 A1 00 00 00 00 00 00 00 00 00 00  .... !..........
 00E0:E5 50 49 50 41 53 4D 20 20 42 C1 4B 00 00 00 01  ePIPASM  BAK....   EXIT Copy Mode
 00F0:A2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  "...............
 Disc is Moonstone XCF2 format.                              Using Drive A:
 ------------------ Viewing Directory Sector ------------------

 ------------------------- COPY MODE --------------------------
  SELECT : █
```

We'll now describe all these options in turn, starting with the control options **A, C, F** and **Z.** These don't do anything directly themselves; rather, they control the exact way the other **COPY** functions work.

The control options **A, C** and **Z** are ***toggles*** - if one is ***ON***, then selcting it will simply turn it ***OFF***, and vice versa. These toggles are all set to defaults which you should ***not*** normally need to change; if you do, then read the following descriptions first!

A Auto Mode

The way that errors are handled during Disc copying operations is controlled by the two toggle options in the **COPY** menu - **A** and **C.**

The **A** command toggles the state of **Automatic Mode** between **ON** and **OFF**.

If Auto Mode is **ON** (which is the initial default setting), then all errors encountered during a disc copying operation are handled automatically by ToolKit.

In addition, any damage to the source disc's configuration record on Track 0 is automatically repaired on the destination disc.

If Auto Mode is **OFF**, then ToolKit prompts for a response to every error detected, as described later under ***Copying a whole Track***.

C CRC Recovery

CRC Recovery controls how one specific type of disc error, CRC failure (reported by the PCW as ***Data Error***) is handled during Disc copies. Unlike all other errors, a CRC error does ***not*** prevent the damaged sector's data from being read. However, it indicates that some of that data (perhaps only a single byte) has been corrupted.

Because of this, Locoscript and CP/M treat the entire sector as being damaged and it is discarded. However, ToolKit will read and display the actual data read from such a sector (along with an appropriate warning). During a Disc Copying operation, this data is usually written to the destination disc exactly as it was read, and a warning message printed on the damage log.

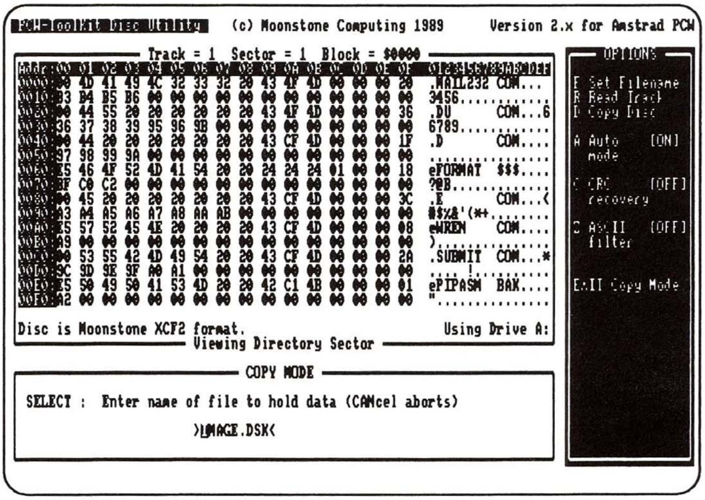
In most cases (such as with sectors holding a file's data), this is the best course of action. However, it could be most dangerous to allow a slightly corrupted directory sector through in this way unless you are aware of the risks and you are able to find and corrupted entries. The CRC Recovery toggle is provided to prevent this.

If CRC Recovery is **OFF** (the default setting), then during a Disc Copying operation, ToolKit treats CRC errors ***in the Directory Sectors only*** as being unrecoverable, and does ***not*** try to read the damaged data. Instead, the destination disc has that sector filled with **$E5** data pattern (used on discs to mark empty directory entries).

If you do want to try recovering a corrupt directory sector, however, then turn the CRC Recovery toggle **ON**. This will force ToolKit's Disc Copying function to recover CRC-damaged directory sectors just as if they were ordinary data. In this case, though, you ***must*** repair the damage to the sector yourself on the copy disc before letting CP/M or Locoscript loose on it - a large, corrupted block number, for example, could damage your disc drive if the PCW tries to move the heads to Track 2000!

F Set Filename

If you select the **F Filename** function, you will be prompted for the name of the file to use to contain data appended from disc using **PASTE** :



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989     Version 2.x for Amstrad PCW

          Track = 1  Sector = 1  Block = $0000
Addr: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000: 00 4D 41 49 4C 32 33 32 20 43 4F 4D 00 00 00 20  .MAIL232 COM...
0010: B3 B4 B5 B6 00 00 00 00 00 00 00 00 00 00 00 00  3456............
0020: 00 44 55 20 20 20 20 20 20 43 4F 4D 00 00 00 36  .DU      COM...6
0030: 36 37 38 39 95 96 9B 00 00 00 00 00 00 00 00 00  6789............
0040: 00 44 20 20 20 20 20 20 20 43 CF 4D 00 00 00 1F  .D       COM....
0050: 97 98 99 9A 00 00 00 00 00 00 00 00 00 00 00 00  ................
0060: E5 46 4F 52 4D 41 54 20 20 24 24 24 01 00 00 18  eFORMAT  $$$....
0070: BF C0 C2 00 00 00 00 00 00 00 00 00 00 00 00 00  ?@B.............
0080: 00 45 20 20 20 20 20 20 20 43 CF 4D 00 00 00 3C  .E       COM...<
0090: A3 A4 A5 A6 A7 A8 AA AB 00 00 00 00 00 00 00 00  #$%&'(*+........
00A0: E5 57 52 45 4E 20 20 20 20 43 CF 4D 00 00 00 08  eWREN    COM....
00B0: A9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  )...............
00C0: 00 53 55 42 4D 49 54 20 20 43 CF 4D 00 00 00 2A  .SUBMIT  COM...*
00D0: 9C 9D 9E 9F A0 A1 00 00 00 00 00 00 00 00 00 00  .... !..........
00E0: E5 50 49 50 41 53 4D 20 20 42 C1 4B 00 00 00 01  ePIPASM  BAK....
00F0: A2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  "...............

Disc is Moonstone XCF2 format.                    Using Drive A:
              Viewing Directory Sector

                     COPY MODE
SELECT :  Enter name of file to hold data (CANcel aborts)

               >IMAGE.DSK<
```

You can now type in any legal CP/M filename. the **DEL**ete key allows you to correct mistakes; press **RETURN** to enter the new name. If change your mind and want to abort at ant point, just press **CAN**cel.

You will then be returned to the **COPY** mode menu; pressing **EXIT** will return you to **COMMAND** mode.

You can change the name of the currently active file like this at any time, as often as you want. Thus you could recover data from more than one file simultaneously, selecting the appropriate recovery file before appending the sector read from disc.

Remember, though, to copy any recovered files out from RAMdisc to a real disc before you reset the PCW or switch it off!

Z ASCII filter

The **Z ASCII filter** toggle in the **COPY** mode options menu controls whether or not data appended is written exactly as read, or is first filtered to ensure it contains no non-ASCII values.

The default setting is **OFF**; this appends data without any modification, and should be used where the exact binary contents of sectors needs to be preserved.

While set **ON**, however, ToolKit will strip the highest-order bit from each byte of data appended, and will then further filter it to ensure it contains no characters other than the standard ASCII set. This consists of the values **32 ($20)** to **127 ($7F)**, along with a restricted range of control codes - **0** (Null), **9** (Tab), **10** (Line Feed), **12** (Form Feed) and **13** (Carriage Return).

Any byte read from disc which is not one of this set is discarded, and an ASCII Null character (**0**) is written to the append file in its place.

This facility is particularly useful in recovering text data from damaged wordprocessor files. The first sector of every Locoscript file, for example, contains a header; if this is lost then the file can still be recovered by appending its remaining sectors form the disc after first turning the ASCII filter ON.

The resulting plain ASCII text file (which will contain your text but no formatting details) can then be read into a new, empty Locoscript document as a Block and resaved properly.

This toggle affects ***only*** sectors appended to a scratch file in RAMdisc from COMMAND mode using the **Paste** key; it has **no** effect during ***any*** other copying operations.

In case you are wondering - **Z** is used for the toggle command as this is the filter parameter used by CP/M's **PIP** utility to do a similar job (and we'd used **A** already!).

Disc Copy Printer Error Log

If a disc error is found during a Track or Disc Copying operation, ToolKit prints a message on the printer describing the nature of the error and the location of the faulty sector.

This record is especially useful in Auto Mode, when ToolKit is handling damage automatically.

There are two types of message. If a sector had a CRC failure then the message

```
*** WARNING *** Source CRC failed at Track x,Sector y in Block $nnnn.
```

With any other type of error, the message

```
Unreadable Bad Sector at Track x, Sector y in Block $nnnn due to <details>
```

will be printed.

Copying a whole Track

ToolKit will also allow you to copy an entire track onto another (or to a different disc).

This capability is useful both for repairing damaged tracks and as a safeguard. If you are going to edit data on a track, you could copy it first into ToolKit's track buffer. That way, you have the original saved as a backup which you can always write back if you make a mess of the modifications!

Note that the track is only held in a buffer for as long as you are in ToolKit. If you exit ToolKit and then re-run it, any data saved will be lost.

If you want to keep a "backup" copy of a track while you return to CP/M to run other programs, then you should write it temporarily to an unused track on the disc, or to a scratch disc kept for that purpose. That way, it is safe and can be re-read and written back elsewhere anytime.

To copy tracks, you must be in **COPY** mode. This is entered by pressing the **COPY** key from **COMMAND** mode.

If you press **R**, the **Read Track** command, ToolKit will read the whole of the current track into its buffer. First, however, you will be prompted for confirmation.

This is a safety measure in case you already have a track held in the buffer and you press **R** accidentally. If you do want to proceed and read the current track into the buffer, press **y** or **Y**. If you don't, press **n** or **N** and the command will be aborted.

One of the most important features of this function is that it allows you to copy a track, ***ignoring errors***.

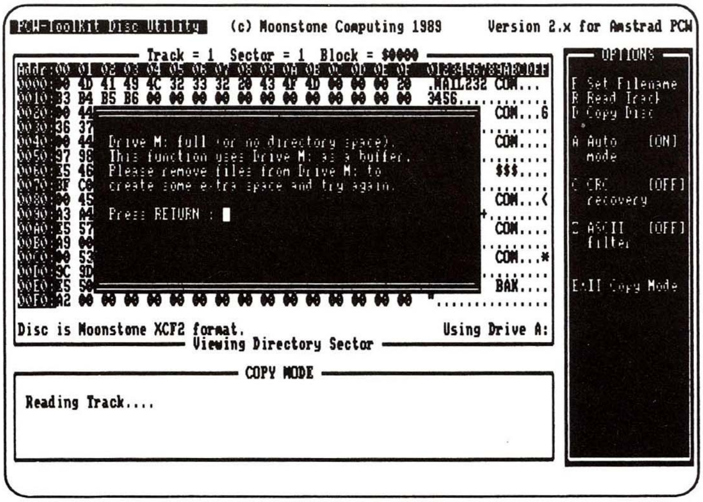
If ToolKit meets a physical error while reading a sector from the track, the usual error window will be displayed (see the scetion entitles Modes and Windows and Prompts at the start of Part Three).

If you select **RETURN**, ToolKit will try to read the sector again, while pressing **EXIT** will abort the track-read operation altogether. If you select **CAN**, however, ToolKit will internally note the existence of the error for use later, but will then abandon that sector and move on to the next one. Thus if only one sector on a track is bad, then only that sector will fail to be copied - you ***don't*** lose the whole track.

Errors will also be listed on the Printer Damage Log.

Track copying effectively operates as if the **Auto** toggle is **OFF**, ***regardless*** of the current setting. The current state of the **CRC Recovery** toggle, however, **is** used.

While copying a track, ToolKit requires a small amount of space (6K) to be free in RAMdisc for its own use. If this is not available, then when the RAMdisc becomes full during the operation, ToolKit will abort with the following Error window being displayed :



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989   Version 2.x for Amstrad PCW

------ Track = 1  Sector = 1  Block = $0000 ------         --- OPTIONS ---
Addr:00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF
0000:00 4D 41 49 4C 32 33 32 20 43 4F 4D 00 00 00 20 .MAIL232 COM...    F Set Filename
0010:B3 B4 B5 B6 00 00 00 00 00 00 00 00 00 00 00 00 3456............   R Read Track
0020:00 44                                                   COM...6    D Copy Disc
0030:36 37                                                   .......
0040:00 44   Drive M: full (or no directory space).          COM....    A Auto      [ON]
0050:97 98   This function uses Drive M: as a buffer.        .......      mode
0060:35 46   Please remove files from Drive M: to            $$$....
0070:BF C0   create some extra space and try again.          .......    C CRC       [OFF]
0080:00 45                                                   COM...<      recovery
0090:A3 A4   Press RETURN : █                               +.......
00A0:35 57                                                   COM....    S ASCII     [OFF]
00B0:A9 00                                                   .......      filter
00C0:00 53                                                   COM...*
00D0:9C 9D                                                   .......
00E0:35 50                                                   BAK....    EXIT Copy Mode
00F0:A2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 "...............
Disc is Moonstone XCF2 format.                       Using Drive A:
------------------ Viewing Directory Sector ------------------

------------------------- COPY MODE -------------------------
 Reading Track....
```

You will then need to exit ToolKit and move and/or delete files from Drive M: to free enough space for ToolKit to use.

When the track has been read, you will now see that the Options window has changed and includes a new option - **W Write Track**. This option will now stay available until you exit ToolKit. You can return to **COMMAND** mode by pressing **EXIT** and select a new track or disc to work on.

At any time after a track has been read into the buffer, you can write the track back out to a disc by pressing **COPY** again to enter **COPY** mode, followed by **W Write Track**.

As usual, ToolKit will prompt for confirmation to avoid accidents! Press **n** or **N** if you have second thoughts and want to abort the write.

If you choose to proceed , ToolKit will first format the target track and will then write each sector from its buffer onto the track in turn. Any errors encountered on the target disc will be reported in the usual Error window, and you can choose to retry, ignore and continue or abort.

If the original track which you read had physical errors and you instructed ToolKit to ignore them, then the equivalent sectors on the target track will be filled with an appropriate data pattern. If the faulty sector is being written to System or Directory areas, it will be filled with the value **$E5**; otherwise, the value **$3F** (the ASCII **?** character) will be used.

Once you have written a track from the buffer, it is ***not*** removed from there, but is retained until you either read another in or exit from ToolKit. Thus you could, if necessary, duplicate a track simply by writing it out as many times as required.

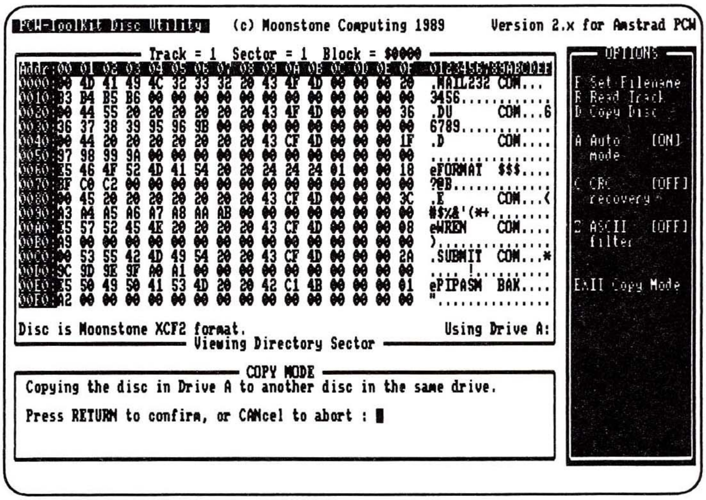
Copying a whole Disc

ToolKit allows you to copy a whole disc, regardless of errors. This is the essential first step to recovering data from a badly-damaged disc, but normal copying programs such as DISCKIT won't do it - they abort if they detect an error, at best discarding a whole track and at worst refusing to continue at all.

To copy a disc using ToolKit, it must first be logged in the normal way. You should then select **COPY** mode by pressing the **COPY** key.

As before, you will see the Options window display the **COPY** mode commands. We are now concerned with the third of these, **D Disc**.

If you select **D**, ToolKit will explain what will happen and prompt for confirmation in the Dialogue window. For example,



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989      Version 2.x for Amstrad PCW

----------- Track = 1  Sector = 1  Block = $0000 -----------      ---- OPTIONS ----
Addr:00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000:00 4D 41 49 4C 32 33 32 20 43 4F 4D 00 00 00 20  .MAIL232 COM...      F Set Filename
0010:B3 B4 B5 B6 00 00 00 00 00 00 00 00 00 00 00 00  3456............     R Read Track
0020:00 44 55 20 20 20 20 20 20 43 4F 4D 00 00 00 36  .DU      COM...6     D Copy Disc
0030:36 37 38 39 95 96 9B 00 00 00 00 00 00 00 00 00  6789............
0040:00 44 20 20 20 20 20 20 20 43 CF 4D 00 00 00 1F  .D       COM....     A Auto     [ON]
0050:97 98 99 9A 00 00 00 00 00 00 00 00 00 00 00 00  ................       mode
0060:E5 46 4F 52 4D 41 54 20 20 24 24 24 01 00 00 18  eFORMAT  $$$....
0070:BF C0 C2 00 00 00 00 00 00 00 00 00 00 00 00 00  ?@B.............     C CRC      [OFF]
0080:00 45 20 20 20 20 20 20 20 43 CF 4D 00 00 00 3C  .E       COM...<       recovery
0090:A3 A4 A5 A6 A7 A8 AA AB 00 00 00 00 00 00 00 00  #$%&'(*+........
00A0:E5 57 52 45 4E 20 20 20 20 43 CF 4D 00 00 00 08  eWREN    COM....     Z ASCII    [OFF]
00B0:A9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  )...............       filter
00C0:00 53 55 42 4D 49 54 20 20 43 CF 4D 00 00 00 2A  .SUBMIT  COM...*
00D0:9C 9D 9E 9F A0 A1 00 00 00 00 00 00 00 00 00 00  .... !..........
00E0:E5 50 49 50 41 53 4D 20 20 42 C1 4B 00 00 00 01  ePIPASM  BAK....     EXIT Copy Mode
00F0:A2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  "...............

Disc is Moonstone XCF2 format.                        Using Drive A:
-------------------- Viewing Directory Sector --------------------

------------------------------ COPY MODE ------------------------------
 Copying the disc in Drive A to another disc in the same drive.

 Press RETURN to confirm, or CANcel to abort : █
```

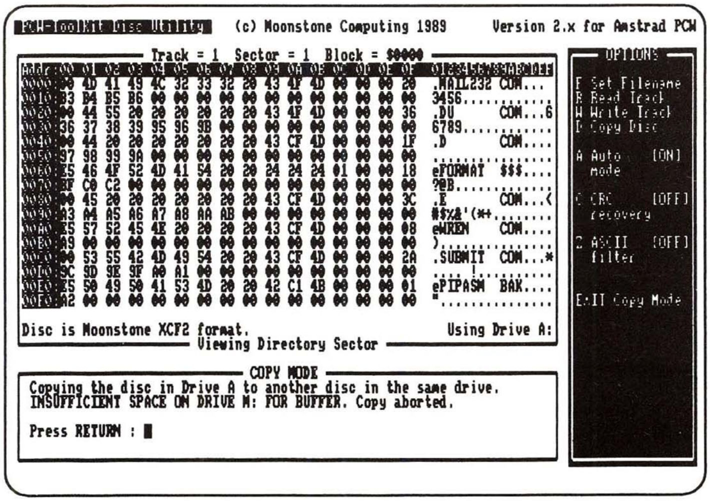
If you don't wish to proceed, press **CAN** and the operation will be aborted. To continue, press **RETURN**.

ToolKit will then prompt you to insert **SOURCE** and **DESTINATION** discs as required, and will proceed with the copying operation.

The way disc errors are handled during the copying operation depends on the settings of the two control toggles **Auto** and **CRC Recovery**, described above.

By default, ToolKit will copy the disc, mark faulty sectors and repair damage to the Configuration Record entirely automatically. A full report of all damage will be printed in the Damage Log, described above.

After selecting the **D Disc** COPY option, ToolKit may instead display the message



```
PCW-ToolKit Disc Utility   (c) Moonstone Computing 1989     Version 2.x for Amstrad PCW

-------- Track = 1  Sector = 1  Block = $0000 --------           ---- OPTIONS ----
Addr: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
0000: 00 4D 41 49 4C 32 33 32 20 43 4F 4D 00 00 00 20  .MAIL232 COM...    F Set Filename
0010: B3 B4 B5 B6 00 00 00 00 00 00 00 00 00 00 00 00  3456............    R Read Track
0020: 00 44 55 20 20 20 20 20 20 43 4F 4D 00 00 00 36  .DU      COM...6   W Write Track
0030: 36 37 38 39 95 96 9B 00 00 00 00 00 00 00 00 00  6789............    D Copy Disc
0040: 00 44 20 20 20 20 20 20 20 43 CF 4D 00 00 00 1F  .D       COM....
0050: 97 98 99 9A 00 00 00 00 00 00 00 00 00 00 00 00  ................    A Auto      [ON]
0060: E5 46 4F 52 4D 41 54 20 20 24 24 24 01 00 00 18  eFORMAT  $$$....      mode
0070: BF C0 C2 00 00 00 00 00 00 00 00 00 00 00 00 00  ?@B.............
0080: 00 45 20 20 20 20 20 20 20 43 CF 4D 00 00 00 3C  .E       COM...<   C CRC       [OFF]
0090: A3 A4 A5 A6 A7 A8 AA AB 00 00 00 00 00 00 00 00  #$%&'(*+........      recovery
00A0: E5 57 52 45 4E 20 20 20 20 43 CF 4D 00 00 00 08  eWREN    COM....
00B0: A9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  )...............    Z ASCII     [OFF]
00C0: 00 53 55 42 4D 49 54 20 20 43 CF 4D 00 00 00 2A  .SUBMIT  COM...*      filter
00D0: 9C 9D 9E 9F A0 A1 00 00 00 00 00 00 00 00 00 00  .... !..........
00E0: E5 50 49 50 41 53 4D 20 20 42 C1 4B 00 00 00 01  ePIPASM  BAK....    EXIT Copy Mode
00F0: A2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  "...............

Disc is Moonstone XCF2 format.                        Using Drive A:
------------------- Viewing Directory Sector -------------------

--------------------------- COPY MODE ---------------------------
Copying the disc in Drive A to another disc in the same drive.
INSUFFICIENT SPACE ON DRIVE M: FOR BUFFER. Copy aborted.

Press RETURN : █
```

and abort. While copying a disc, ToolKit requires scratch space to be available in RAMdisc; the amount of space required varies depending on exactly what type of disc is being copied.

If this minimum amount of space isn't free, then the operation can't proceed. You will first have to exit from ToolKit and move and/or delete some files from the RAMdisc to free enough space for ToolKit's buffer.

PART FOUR

Technical Reference

This section describes the more technical aspects of PCW disc formats. You should ***not*** require to know this for most of the work you are likely to do with ToolKit; however, more complex operations such as re-building a damaged directory will require a fuller understanding of the following information.

Disc Directory structure

As we saw in Part One, in order to impose a useful data structure on a disc the PCW puts our data into ***files***. These files can be located ***anywhere*** on the disc - the data blocks within a file need not even be laid out contiguously. Data can be put into ***any*** free data block; the PCW keeps track of where each one is, and the order in which they were used, in the ***directory*** for the disc.

On all PCW discs, the very first track (Track 0) is reserved and isn't used for storing anything, and so the data space on the disc starts on the next track (Track 1).

The Directory occupies the first few blocks at the start of the data space; obviously, the space it occupies is reserved and can never be used for holding our data.

The exact number of sectors and blocks used to hold the directory, and their locations, for the various types of PCW discs are as follows :

Disc Type	Num Dir Entries	Num Secs	Blk Size	Blks Used
CF2	64	4	1K	0,1
XCF2	96	6	1K	0,1,2
CF2DD	256	16	2K	0,1,2,3
XCF2DD	256	16	4K	0,1

Every file on the disc has one or more directory entries associated with it. Each directory entry is 32 bytes in size, and contains information such as the name of the file, its size and its location on disc. A "dump" of a typical directory entry is shown below.

```
0534944202020 20  20434F4D0000003A   .SID    COM...:
33435363738393A   0000000000000000   3456789:........
```

This dump shows 16 bytes (in Hex) on each line, followed by the ASCII character which corresponds to each value if one exists. If there is no ASCII character for a value, a "." is shown. The ASCII dump is useful to see where the filename (SID.COM, in this case) is stored.

User / Group Number

The first byte contains the ***User Number*** associated with the file; this can be a value from 0 to 15 ($0F). CP/M uses a value of 0 by default for all files; this can be changed by the USER command but on small systems such as the PCW this is best forgotten!

Locoscript makes much more use of user numbers, in order to divide files into ***Groups*** and provide the ***Limbo*** facility. Each disc can have up to 8 Groups (0 to 7), and each Group has associated with it a ***Limbo Group*** where deleted files are held before being physically deleted, allowing you to change your mind! The Limbo Group associated with a file Group has the User Number of that Group plus 8.

So, for example, a file deleted into Limbo from Group 0 will have its user number changed from 0 to 8, while a file in Group 4 would have it changed from 4 to 12 ($0C). Note that Limbo files are ***not*** physically deleted as far as the PCW is concerned. Limbo is just a feature of Locoscript which uses a different User Number to simulate a "halfway house" for files you want rid of.

A user number value of **$E5** means that the file ***has*** been physically deleted. Note though that when CP/M ERAses a file, it doesn't actually "wipe out" the contents of the file on disc - it simply sets the user byte in all the file's directory entries to $E5 as a marker.

CP/M will then re-use the data blocks belonging to the deleted file as and when it needs them. It is only once the blocks have been re-used for something else that their data is lost, so you **can** recover accidentally deleted files - ***provided nothing new has been written to the disc in the meantime*** - by simply changing the user number in all of its directory entries from $E5 back to a value between 0 and $0F.

Filename

Bytes 1 to 8 contain the filename in ASCII upper case; bytes 9 to 11 hold the file type, again in ASCII upper case. Note that the "." we normally put into "filename.typ" is ***not*** put into the directory.

You may find that some files have one or more of the letters of the filename extension (Bytes 9 to 11) modified by having the top bit of the byte set. This adds 128 ($80) to the value of the character, and so it will not display as the letter it actually represents.

This is done to "tag" the file with special attributes, such as read-only or system; pressing the **RELAY** key toggles whether or not this bit is ignored when the ASCII equivalent is shown.

Bytes 12 and 15 are the ***extent number*** and ***record count*** values respectively; we'll return to these later. Bytes 13 and 14 are always 0; they are reserved for internal use by the PCW during file operations.

Block Index

Bytes 16 to 31 are the "index" which tells us where to find the data for all or part of the file - each byte is the number of a ***block*** which has been allocated to the file. For example, a CF2 disc uses a 1K block size; each directory entry has 16 bytes reserved for holding allocation information and so can "map" up to 16K of a file. If the file is 16K or less in size, then obviously it only requires one directory entry - we say it has a single ***extent***, and its ***extent number*** (byte 12 of its directory entry) is set to 0.

An ***extent*** is just another word for a 32-byte directory entry; in general we refer to a file as having "a" directory entry (in the sense that its name only appears once when we list the directory using CP/M's **DIR** command). However, we have just seen that a single 32-byte extent can only "map" the position on disc of up to 16K of data. If a file is larger than 16k, then it will need more than one extent to hold its directory information.

Record Count and Extent Number

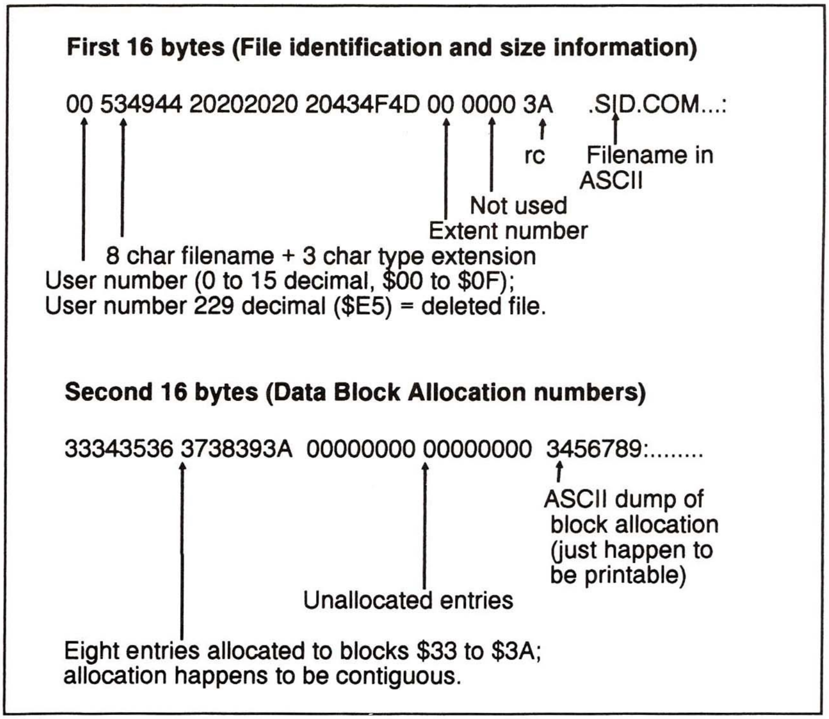
The ***Record Count*** (byte 15) is the exact number of records which contain valid data held within the blocks pointed to by an extent. As an extent can index up to 16k, and a record is 128 bytes in size, then the maximum number of records held by an extent is 16 x 8, or 128 ($80). Thus extents of large files which are completely filled will have record counts of **$80**; the last extent of a file, if it is not completely filled, will have a record count less than this.

When a file requires more than one directory extent, as many 32- byte (16K) directory extents are allocated to the file as are needed to handle the whole file. For example, a file which is 46K long will require three extents to hold its directory information. The ***extent number*** (byte 12) in the first extent is set to **0**, that in the second extent is set to **1**, and that in the third extent is set to **2**.

Also, because the first two extents have been completely filled in this example, ***all*** sixteen of their allocation numbers will be used and their ***record counts*** (byte 15 of each) will be set to 128 ($80). The third extent will contain the allocation information for the remaining 14K of the file, and so will have the last couple of allocation numbers free.

The record count byte of the third extent will contain the exact number of 128- byte records which are referenced by the blocks in the extent (the last block allocated may ***not*** be full, though any spare space within it is still allocated and cannot be used for storing anything else).

The following diagram shows the example directory entry for the simple, small file used previously (less than 16K in size, and so requiring only one extent) in more detail, and may help make things clearer:



First 16 bytes (File identification and size information)

```
00 534944 20202020 20434F4D 00 0000 3A    .SID.COM....:
```

rc
Filename in ASCII
Not used
Extent number
8 char filename + 3 char type extension
User number (0 to 15 decimal, $00 to $0F);
User number 229 decimal ($E5) = deleted file.

Second 16 bytes (Data Block Allocation numbers)

```
33343536 3738393A 00000000 00000000 3456789:........
```

ASCII dump of block allocation (just happen to be printable)
Unallocated entries
Eight entries allocated to blocks $33 to $3A; allocation happens to be contiguous.

Extent Folding

The discussion above assumed that the disc has less than 256 data blocks, as we were mapping each block into a single byte in the Index part of each directory entry.

While this is the case for small capacity discs (such as single- sided 40 track PCW CF2 and XCF2 discs), the situation becomes more complex with larger discs such as CF2DD and XCF2DD.

Due to restrictions which will not be discussed here, the maximum number of blocks a PCW can accommodate on any disc is 364. Thus, as CF2DD discs have a capacity of over 700K, it is obviously impractical to use a block size of 1K on such discs. Instead, a block size of 2K is used. Even so, this still results in such a disc having more than 256 blocks.

To allow all these blocks to be mapped, we have to use 2-byte allocation numbers in the Index. This means that each extent can now only map 8 blocks instead of 16 (half as many as each entry is now 2 bytes instead of 1), but that each block holds twice as much data (2K instead of 1K). Thus, each extent on a CF2DD disc still indexes a total of 16K of data from a file.

The situation with XCF2DD discs created with XFORMAT is however even more complex! As XCF2DD discs have a capacity of 800K, using a 2K block size (like standard CF2DD) would result in the discs exceeding the 364-block maximum limit imposed by the PCW.

We therefore use a 4K block size on such discs, resulting in just 200 blocks. Thus these discs can revert to using single- byte allocation numbers in directory extents, and so each extent can reference up to 16 blocks. Now, though, each block is 4K in size and so each extent can index up to 16 x 4 = 64K, four times that of a normal CF2, XCF2 or CF2DD extent.

The CP/M term ***Extent Folding*** is used to describe this situation, and a fuller description of the effects (on Record Count and Extent Number values, for example) is beyond the scope of this manual. Please refer to an appropriate CP/M reference for more details on this if it is required.

PCW Disc Configuration Record

All PCW discs have Logical Track 0 reserved, and this is used to hold a ***Configuration Record*** and ***Boot Sector***. The Boot Sector is used only on "Start of Day" discs, and holds the short program required to read the large .EMS system file into memory and execute it.

The Configuration Record is used on all discs to allow the PCW to automatically detect the format of any disc used. This is how the PCW always knows the type of disc in use without having to be told by the user.

In practice, this feature is not as powerful as it first appears. The range of formats which the system can accommodate is fairly limited, as Descriptor Blocks are checked by the BIOS to ensure that certain of the parameters fall within fairly rigid limits.

The disadvantage to the system is its vulnerability. If this small record is damaged or altered accidentally, then the PCW will be normally totally unable to use the disc - even if the rest of it is completely healthy! This even prevents the use of "standard" CP/M disc repair tools such as DU.COM.

ToolKit recognises all standard format parameters for discs formatted with DISCKIT and both versions of XFORMAT. It ***can*** however also work with discs whose Configuration Record is missing or corrupt, and can easily and automatically fix the disc by writing an appropriate Configuration Record.

You should never need to modify a Configuration Record manually, and indeed should **never** do so unless you know exactly what you are doing.

The details of the contents of the Configuration Record are really only of use for people who require to use an unusual format and a fuller discussion of them is outwith the scope of the ToolKit manual! If you have our MFU software, you will find a large amount of detail on all topics of disc formats in the technical appendices to the MFU User Guide.

The contents of the Configuration Record (also known as a Format Descriptor Block) are shown below. Please note that this block is **not** documented by Amstrad, and so the following description is based entirely on our own observations and deductions. Consequently, we can accept no responsibility for its accuracy!

Byte 0 : Disc Type - 0 = Single Sided
3 = Double Sided

Byte 1 : Sidedness - 0 = SS
1 = DS Cylinder Access
2 = DS Side Access

Byte 2 : Tracks per Side

Byte 3 : Sectors per Track

Byte 4 : FDC "N" (log2(Sector Size) - 7; = DPB "PSH")

Byte 5 : Reserved Tracks

Byte 6 : log2(Block Size) - 7

Byte 7 : Number of Directory blocks

Byte 8 : Gap (R/W)

Byte 9 : Gap (Format)

Bytes 10 - 15 ($0A - $0F) : Reserved (appear to be **$00**)

Bytes 4, 8 and 9 are used in programming the uPD765 Floppy Disc Controller (FDC) chip inside the PCW; further information on these can be obtained from the appropriate NEC data sheets for this device.

Do ***not*** alter any of these values on a working disc or you risk corrupting all the data held on it.

PCW Disc Error Messages

As any PCW user is painfully aware, a disc fault is accompanied by one of a variety of equally obtuse error messages! This section attempts to explain what each of these actually mean, as well as suggesting possible causes and remedies.

Errors detected by ToolKit itself are handled rather more fully than by CP/M or Locoscript. In addition to simply showing the appropriate error message, ToolKit also displays the FDC Status and ID Result bytes received from the Disc Controller.

This information gives you the "bottom line", if you know how to interpret it. Obviously, this will only be of interest to programmers who are familiar with the function of the uPD765A, but it can then be invaluable. If you ever wish to report a problem to us which generated a disc error message, these results bytes will help us enormously to determine what caused your fault.

Hardware faults on the disc drives themselves are relatively uncommon, but do happen. The detailed descriptions given below of the various error reports also show which may be caused by hardware faults.

If you suspect that you may have such a fault, then ***don't*** write to important discs with that drive, as this may destroy all the data on them! Test it by ***reading*** from discs which are known to be OK (such as your original CP/M and Locoscript discs), and by formatting and writing to ***blank*** discs and testing these, if possible, on a different computer or drive.

The commonest hardware faults are due to wear or slippage of the stepper motor. This controls the exact positioning of the read/write heads over the different tracks, and if a drive becomes even slightly misaligned it will no longer be able to accurately read and write properly-formatted discs.

It may, however, be quite happy to use discs it has itself formatted recently, so disaster may not strike until you try to use an older, differently-aligned disc! It is worth using DISCKIT occasionally simply to VERIFY an old and trusted disc (such as a distribution disc) just to check that the drive isn't drifting out of alignment.

Drive not ready

The disc drive reported that it was not ready to transfer data when it should have been. Usually, the cause is simply that there isn't a disc in the drive, or it hasn't clicked home fully when you inserted it! If this is not the case, then you may have a fault in the disc drive or its cabling - if the error recurs, then seek technical advice.

Disc Write Protected

This error means that the FDC has attempted a WRITE DATA operation and the disc drive has reported that the disc has its Write Protect slider pushed in.

Seek Fail

The FDC has stepped the heads to the requested track, but has failed to read the corresponding Track ID information from the disc.

This is a "composite" error constructed by the BIOS from several low-level reports from the FDC chip, and should not occur on a PCW under normal circumstances.

If you are using format-altering software such as MFU, then the cause is usually that the track pitch of the disc is not the same as the FDC has been told! Otherwise, this error means a serious fault is present on either the disc or the drive.

See also ***Missing Address Mark***.

Data CRC failure

The sector which the FDC has just read from the disc is corrupt. Each sector on the disc has a complex polynomial checksum added to it when it is written, calculated from all bytes of its data. If one or more of these bytes then changes, the checksum will no longer validate when the sector is read.

Occasionally, the checksum itself may become corrupt rather than the data, but this is extremely unlikely statistically.

Usually, the only remedy is to copy off all the files which are readable, and reformat the disc. Using ToolKit, however, you will be able to see what data was actually returned from the disc, and determine the extent of the corruption for yourself.

Frequent CRC failures on different discs, or on apparently random sectors throughout a disc, suggests either a drive fault or, more likely, that your drive heads need cleaned!

No Data

The FDC has been unable to find a sector whose ID Record matches the Head, Cylinder, Sector and N values specified in the FDC command. This will happen if any of these specified values is incorrect (for example, if Sector 0 on a track has been asked for when the first sector number is 1). This should never occur unless you are using format-changing software such as MFU. Otherwise, it suggests a fault in the disc drive.

Missing Address Mark

This notorious message means that the FDC has failed to find the Sector's ID Address Mark (IDAM) - i.e. it cannot find the sector anywhere on the track at all - **or** the sector's Data Address Mark (DAM) cannot be found. Either of these can occur if the disc has become heavily corrupted (perhaps by a magnetic field, or by physical damage), **or** if the track being read is simply unformatted!

If the error is not simply due to an unformatted track or disc, inspect the disc for physical damage by turning it slowly by hand and looking at both sides of the Read/Write slot. If there appears to be no sign of damage, try reformatting.

Uncommonly, this error can be induced wrongly by unusual discs which have a ***Deleted Data Address Mark*** (DDAM) instead of the normal DAM. This special marking was used in the past on larger computers to tag deleted data and is uncommon on micros. DDAMs should never occur on standard PCW discs, but are sometimes used as a crude copy-protection mechanism.

If this error occurs frequently with a number of different discs, it suggests a fault in the drive.

FDC data overrun

This means that the main system has failed to service a request to read data from the FDC quickly enough, and the data has been overwritten and lost within the FDC. This should never happen unless either a fatal software or hardware failure has occurred.

Bad Format

This is a "fake" error produced by the PCW if it is deeply unhappy with life, and need not mean that there is anything wrong with the disc. The literal meaning is that the PCW could not determine the format of a disc, and suggests that the Configuration Record on Track 0 may be damaged. However, you may find that the machine gets stubborn and produces this error with ***any*** disc you try - in that case, reset the PCW and try again; there may actually be nothing wrong with the original disc.

If the error is genuine, the cause may be corruption during a disc copying operation, where part of the source disc has accidentally been copied back onto itself. This problem, and the procedure to cure it, is described in Part Two.

Media Changed

"Fake" disc error generated by BIOS. When the BIOS tried to read a track, it found that the sectors on the track had different numbers to those specified in the BIOS internal XDPB data area. Again, this is unlikely to appear in normal use as it is a consequence of altering format parameters within the PCW with programs such as MFU.

Disc unsuitable

This is another "fake" error code produced by Amstrad's BIOS. This can occur on a PCW8512 if you try to write to, or format, a 40 track disc in Drive B, or on a PCW9512 with any attempt to write to a 40-track CF2 or XCF2 disc outwith ToolKit.

Notes

Notes

MOONSTONE COMPUTING
Strathclyde Business Centre
31 Clyde Street
Clydebank
Glasgow G81 1PF
Tel: 041-941 3120