MFU

Version 4

User Guide

MOONSTONE
COMPUTING

MULTI - FORMAT UTILITY SOFTWARE

Version 4

USER GUIDE

AND

TECHNICAL REFERENCE MANUAL

COPYRIGHT NOTICE

Published by Moonstone Computing 1987

Printed in the U.K.

CONTENTS

1. INTRODUCTION

Welcome to the Users' Guide and Technical Reference Manual for Moonstone Computing's Multi-Format Utility (MFU). When you run MFU on an Amstrad PCW8256 or CPC464/664/6128 equipped with a 5¼" external disc drive, you instantly have one of the most powerful Multi-Format systems available anywhere - at a fraction of the multi-thousand-pound prices of other machines available to do a similar job.

If you have also bought one of our 5¼" disc drives, please read the other book in the package - the Hardware Installation Manual - before proceeding any further here. This will guide you through attaching and testing your disc drive and/or RAM upgrade. The Hardware Installation Manual is also available either separately or with cabling if you already have, or wish to acquire, a third-party disc drive.

This book is designed to guide you through the process of installing and using the MFU software. The next section deals with copying of your master disc to create a working copy, as well as covering installation and customisation to suit your own requirements. Following that, there is a general but comprehensive guide to firstly the concepts involved in disc formats, and secondly the CP/M operating system as it affects us.

Following this, we describe in detail the various ways MFU can be used, covering all the options from analysing unknown discs to formatting new ones or transferring data between your Amstrad and other computers.

At the end of the manual, there is an extremely comprehensive technical reference section, containing several detailed Appendices. These give in-depth information on several topics which were covered more generally in the main part of the manual for those who require this level of information. Topics include the way the Disc Controller chip writes information on a disc, CP/M's internal Disc Parameter data structures and disc Directory formats, as well as a full explanation of the various disc errors which can occur, and how to remedy them. Details of the hardware modifications required to use Single Density discs with a PCW8256/8512 are also given.

So what does MFU do?

Using the MFU system, you can read and write data on virtually **any** type of soft-sectored 5¼" floppy disc.

MFU's functions fall into three main groups :

1. Drive B can be set up to exactly emulate almost all types of CP/M formats. This means that you can use a disc (even run programs which are on it!) from another CP/M computer on your Amstrad. Once a format is set up, you can read, write and even format discs which the other computer will also be able to use.

   This instantly gives you access to the huge amount of existing CP/M software which is available, but not on Amstrad format discs! If you use other CP/M computers either at home or at work, you can use the **same discs** in them and your Amstrad.

   MFU includes a comprehensive library, containing over two dozen common CP/M formats (such as Superbrain, Wren, Osborne, Epson and DEC Rainbow), so **no** expert knowledge is required to emulate and use such a format. However, if the format for a computer isn't on the list, you can define it yourself and add it to the library for future use.

   Once a format has been loaded, **no** further software is required to use it - you can leave MFU and your computer will treat Drive B **exactly** as the other computer would. Any programs you want to run will know that Drive B is emulating the other format and will read and write data correctly.

2. It is not possible to emulate non-CP/M formats in the same way, unfortunately. Different computers use different ways of storing data on disc, and programs from one will not in general run on any other. However, MFU lets you transfer files between CP/M and a variety of other computers, including IBM-PCs and compatibles, and BBC and Master computers.

   Thus you can easily transfer your data around between different types of computers on disc, getting round the traditional problem of different computers being completely incompatible with each other.

   So, for example, you could write a letter on a BBC at home, use your Amstrad and MFU to transfer it to an IBM-PC at work for printing on the office laser printer - and then copy it onto a disc which your secretary can use in her Superbrain CP/M machine!

3. For the expert, MFU allows low-level analysis, inspection and alteration of the data and programs held on virtually any 5¼" disc, even those which MFU's higher-level emulation and file transfer functions cannot support.

   MFU has the ability to automatically analyse and determine the physical format of any disc, including density, number of sides, track pitch, number of sectors on a track and their size. Physical sectors can be read, edited and written in either hex or ASCII using the built-in sector editor.

   It is also possible to create physical copies of almost any type of disc, again including formats which the higher-level functions cannot handle.

MFU has been specially written for Amstrad computers to take full advantage of the capabilities of their Floppy Disc Controller (FDC) chip, and of Amstrad's CP/M Basic Input/Output System (BIOS). For example, when a 40 track format is installed and your drive has 80 tracks, MFU will automatically instruct the BIOS to "double-step" the heads - this means that you **don't** need a switchable drive to use both 40 and 80 track formats!

Throughout this manual, all topics will be presented in as simple and straightforward a manner as possible. MFU is designed to be simple and friendly to use, and you do **not** need any detailed knowledge of CP/M before you can install and use formats from the library. Obviously, however, some of MFU's functions (such as editing the format parameters, for example) **do** require a good working knowledge of the meanings and functions of these parameters, and readers are referred to appropriate parts of the Technical Appendices where these are relevant.

Note for CPC464/664/6128 owners :

Please note that throughout this manual, we refer to the **RETURN** and **EXIT** keys, as found on a PCW. For a CPC version of MFU, the corresponding keys are **ENTER** and **ESCape** respectively.

UPGRADE POLICY

We are continually improving and enhancing MFU, and it is our policy to issue periodic updates to existing customers. A small handling charge will be involved to cover the costs to us of manuals and postage, but upgrades will otherwise be **FREE** of charge if the previous master disc is returned to us. All existing customers are notified automatically by us when major upgrades are released, but if you find a bug or have a difficulty you can't understand please contact us - it may be that someone else has reported the same problem and a revision is available.

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.

2. INSTALLING MFU.

In general, **MFU requires no special installation before it can be run.**

However, for technical reasons it is only possible for us to support BIOS version 1.4 (for a PCW8256/8512) or 1.0 (for a CPC464/664/6128).

You can determine your BIOS version by looking at your computer's sign-on message when you boot.

If you have a different version of CP/M, MFU will NOT run.

If you find that you have an **earlier** version of CP/M, you should contact Amstrad themselves and request an upgrade of your CP/M and LocoScript disc to the current release; they will do this for you **free of charge. Please do NOT ask US for an updated CP/M - we CANNOT supply it!**

If you find that you have a **more recent** version of CP/M, please let us know - we'd like to hear about it!

IF you are using a switchable drive, please note the following:

If your version of MFU is set up for an 80 track drive (it WILL be unless you have requested a special version), ALWAYS have your drive set to "80 track" mode - EVEN when using a 40 track format. MFU will adjust the system to double-step automatically whenever required.

2.1 Distribution files

The current version of MFU has a number of files which are necessary for its full operation; these are as follows :

MFU.COM	- This is the main program; to run the system simply type "MFU" at the M> prompt.
MFU.DAT	- This is MFU's library file, which contains details of all its stored formats. This is distributed with a large selection of common formats, but you can alter these or add to them easily from within MFU.
MFU.000 to MFU.009	- These are machine code overlay files to which are required by MFU to perform its various functions.
MFUPATCH.COM	- This is a special "patcher" program which allows you to customise your working copy of MFU. You can alter any of the system defaults and specify printer and screen control codes.

Use CP/M's

A>**dir**

command on Side A of your distribution disc to make sure that all the files are present.

PCW versions ONLY :

With PCW versions of MFU **only**, we also supply a disc formatting utility called **XFORMAT**. This is a replacement for Amstrad's **DISCKIT** utility which formats blank discs both faster and to a much higher capacity! Using XFORMAT instead of DISCKIT will give you an extra 20K of space on a CF2 Drive A disc and an extra 80K on a CF2DD format disc on Drive B (including 5¼" versions, obviously!).

The XFORMAT program is contained in a single file, named

XFORMAT.COM

on Side B of the distribution disc.

Instructions for using XFORMAT will be found in a separate booklet.

2.2 Backing up your distribution disc.

Please, **before** you do anything else, make a working copy of your MFU distribution disc! Then put the original safely away to prevent it becoming corrupted or damaged in use. **NEVER** use the master copy of the software as the working version - sooner or later you will have a very frustrating (and expensive) disaster, as we cannot supply replacement discs in the event of such an accident.

PCW versions:

Please note that we now distribute PCW versions of MFU on **High Density** 200K 3" discs, formatted using our **XFORMAT** program. These discs are perfectly compatible with normal CF2 discs (except for their greater capacity) in all respects bar one - Amstrad's **DISCKIT** copying program **CANNOT** copy them!

Consequently, you must copy the distribution disc EITHER using a **FILE** copying program such as **PIP.COM** and **NOT** a **DISC** copying program such as **DISCKIT**, OR by using XFORMAT's **C**opy disc option. Details of the latter method are given in the XFORMAT User Manual.

To copy the MFU disc file-by-file, put your CP/M disc containing the PIP program into Drive A and type

A>pip

Now put your MFU distribution disc into Drive A and type

***m:=a:mfu*.*[v]**

This will copy all the files off the distribution disc into the RAMdisc (Drive M:). Note that Drive M should be **empty** before you start this operation or you may run out of room!

Once you have done this, put a blank, **FORMATTED** disc into Drive A and copy the files back onto it by typing

***a:=m:mfu*.*[v]**

Note that the **[v]** option tells PIP to verify each file after it has been copied to ensure none has been corrupted.

What to do if you run out of memory!

If you only have a 112K RAM-disc, you will **not** have enough space in it to hold **all** of the MFU files at once and the copying procedure we described above will abort with the message

ERROR: DISK WRITE NO DATA BLOCK - <filename>.$$$

when all of the space in Drive M has been filled.

If this happens to you, **don't panic!** Firstly, copy the files which **are** in Drive M out onto a blank 3" disc in Drive A as described above.

Once those files are all safely copied out, note down their names. Check against the list of files given above to determine which have not been copied.

Type the command

A>era m:*.*

followed by "**y**" when CP/M asks for confirmation. This will wipe Drive M clean again. Now type

A>pip

again and put your distribution disc back into Drive A.

You will now have to copy each of the files you **didn't** get the first time from Drive A into Drive M **individually** by giving a series of commands like

***m:=a:<filename>.<ext>[v]**

where **<filename>.<ext>** is the name of each file to be copied in turn.

This will get all the extra files into Drive M. Now put your copy disc into Drive A and again type

***a:=m:*.*[v]**

as before, to copy the remaining files out of Drive M onto the copy disc. Once this is done, press <RETURN> to get out of PIP and type

A>dir a:

to check that **all** the files are now present on your copy disc.

CPC versions :

If you have a version of MFU for a CPC464/664/6128, the procedure is much more straightforward! Just use Amstrad's **DISCKIT3** program to make a backup of the whole disc.

2.3 Customising MFU

Version 4 of MFU is pre-configured with several default options. These include parameters describing the type of disc drive supplied with MFU (or that with which it is expected to normally be used), and the speed tolerance of that drive as a percentage as well as terminal and printer control strings, and the designation of your RAMdisc, if you have one.

Some of these values can be altered temporarily from within MFU with the **S)** option; if you wish to alter them permanently (if you later buy a different drive, perhaps), then this can be done by patching MFU.COM itself using MFUPATCH.COM.

Full information on customising your copy of MFU is given in Appendix A.

3. USING MFU.

To run MFU, the computer must already be running CP/M. If you are still in AMSDOS on a CPC, or Locoscript on a PCW, then put your CP/M boot disc into Drive A and type **CPM** (on a CPC) or press the **SHIFT**, **EXTRA** and **EXIT** keys together to reset the computer (on a PCW). This will start CP/M running. When it is ready to receive a command, you will see the prompt

A>

To execute MFU, make sure you have all the files which make up the MFU system on the logged drive, then type **mfu** at the CP/M prompt.

You can also execute MFU from a SUBMIT file if you wish.

The following sections provide information on restrictions on the ways you can use MFU, as well as suggestions to make its use faster and easier.

3.1 Drives and Files

If you have a PCW8256, we suggest that you run MFU from the RAMdisc rather than from Drive A as this will speed up its operation greatly. All further discussion will assume that MFU and all its files are present in Drive M, and that M: is logged as the default drive (i.e. your CP/M prompt is **M>**).

CPC6128 owners **must** run MFU from Drive A unless they have a third-party RAMdisc upgrade - operation will be slower, but otherwise identical. Whichever drive contains MFU, it **must** be made the default drive before MFU is run or the program will fail to find its overlays. This can be altered by MFUPATCH; see Appendix B for details of this.

If you are running MFU from Drive A, **NEVER** remove the disc containing the program and its overlays from the drive while MFU is in use unless prompted to do so. Doing this will result in a non-recoverable error and MFU will abort.

NEVER attempt to run MFU from a disc in Drive B unless you are sure you know what you are doing; if you do this then change the format assigned to Drive B, MFU will not be able to find its library files on disc and will be unable to function fully.

If you are using a 40/80 track switchable disc drive, **always** keep it set to **80 track** mode, even if you are using a 40 track format. MFU will alter the BIOS to automatically compensate for the different track pitch.

The **E)**dit and **A)**nalyse functions both make use of the cursor keys to provide on-screen editing of parameters and data. For this to function properly, MFU sets the cursor keys up to supply WordStar-compatible codes.

If you want to restore CCP cursor keys on exit from MFU, we suggest you run MFU from a SUBmit file such as the following :

```
mfu
setkeys keys.ccp
```

We also suggest that you load the MFU system into the RAM-disc by including the necessary commands in your PROFILE.SUB file which the computer executes automatically when CP/M is started up.

The following sequence of commands will load all necessary files from a 3" disc in Drive A: into RAM-disc, change the logged drive to M: and execute MFU:

```
pip m:=a:mfu.*
m:
mfu
```

Add these to your PROFILE.SUB file using RPED (or your favourite text editor) if you want CP/M to execute the sequence for you automatically on boot.

Please note that the whole MFU system requires 140K of disc space; an unexpanded PCW8256 only has 112K available in Drive M and so on such a machine you will have to run the program from Drive A, with a corresponding reduction in speed of operation - or buy a RAM upgrade!

However, if you want to cut down on the space taken up by MFU on your working discs, the following list of files explains the function of each. If you are not intending to use one or more of the functions, the corresponding files need not be present.

MFU.COM	-	Main program; **must** be present.
MFU.000 to **MFU.004**	-	Main overlay files; **must** be present.
MFU.005	-	Only required by the **P)** option.
MFU.006, **MFU.007**	-	Only required by the **N)** option; used for CP/M <-> MS/PC-DOS file transfer.
MFU.008, **MFU.009**	-	Only required by the **N)** option; used for CP/M <-> BBC file transfer.
MFU.DAT	-	Default library data file; must be present if you want to use formats stored in it or for MFU to automatically display the format in use.

When MFU is first run (and at frequent points within the program) an attempt is made to identify the Drive B: format currently in place by referencing the Library file. For this facility to be of use, MFU.DAT **must** be on the logged drive. If the current format doesn't match any in the library then the message

Drive B: is currently set to an unknown format

is displayed at the bottom of the screen. Otherwise, the format is named.

Please note that the displayed format is that which is currently set up **in memory** for Drive B, and need **not** bear any relation to the **actual** format of a disc you may have put into Drive B. If you put a different type of disc into Drive B, you must also load the appropriate format from the library to allow MFU and CP/M to understand it!

If you don't know the format of disc, you can use the **A)**nalyse option from MFU's main menu. Section 6.1 explains how this option works.

3.2 Memory Restrictions

MFU reserves **almost all** of the TPA (from **&100** to **&EFFF** inclusive, to be exact) for its own use. This means that if you have any RSXs loaded **below** &F000, MFU may overwrite them and the BDOS will probably crash when any RSX-using program is run. MFU will attempt to determine the amount of memory available using standard legal BDOS calls, and will abort if the apparent TPA top is below &F000.

4. STRUCTURE OF A DISC

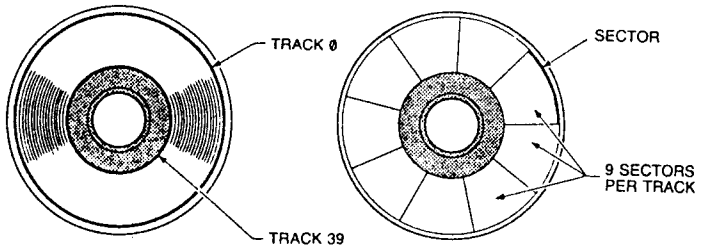
In this section, we will explain the basic facts behind the use of discs to store data on computers, and look at the restrictions physical disc structure puts on our format definitions. If you are already familiar with these concepts, you may wish to miss out this section. A much more detailed description of Physical Track formatting is given in Appendix B.

4.1 Basics

A floppy disc is simply a disc of plastic, coated in the same sort of magnetic material as an audio or video tape. Before a computer can use a disc, it must be **formatted** - this process writes a preset data pattern all over the disc, dividing it up into concentric circles, called **tracks**, within which our files will be stored.

This is what is meant when a disc is described as "40 track", or "80 track" - this simply describes the number of tracks of information which the disc can hold. Similarly, an "80 track" disc drive is a drive unit which is calibrated so as to be able to position its read/write heads over 80 concentric tracks. It is more accurate technically to use the term **96 tpi** (Tracks Per Inch) when referring to "80 track" drives and disc formats, as some drives and formats do not use exactly 80 tracks on the disc! Similarly, "40 track" drives and formats are better referred to as **48 tpi.**

All tracks on a disc are the same width and are spaced the same distance apart, so a disc drive can read or write any track on the disc by **stepping** its heads in or out across the disc by the appropriate amount.



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.** These are simply a way of dividing the storage space on a disc into convenient little blocks, which the computer can keep track of. 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 checksum to verify any data which is stored in it.

4.2 Track Pitch

As you may have realised, the tracks on a 48 tpi disc are spaced approximately twice as far apart as the tracks on an 96 tpi disc. This is not normally a problem if you have an 80 track disc drive; MFU can automatically "double step" the heads to compensate for the higher track spacing when using a 40 track disc. However, there is a theoretical problem inherent in using an 80 track drive to write data onto 40 track discs.

This is due to the fact that not only are the tracks on 48 tpi discs spaced further apart, but they are also **wider** than on 96tpi discs. In general, the read/write head on a 48tpi drive is larger than its 96tpi equivalent; this is largely just because they tend to be older and less advanced technologically.

This can have an effect when a 96tpi head writes data onto a track which **already** has 48tpi data recorded on it. It is possible that while the 96tpi drive will happily read its own data back again, the wider head on the 48tpi drive will **also** read some of the surrounding remnants of the old, wider data track with the result that the data read becomes garbled or corrupt.

While this must always be borne in mind when using a 48tpi disc in both types of drive, in practice it is rarely a problem. We ourselves have in fact yet to find a 48tpi drive which **does** have problems reading such a disc! We regularly use MFU and a variety of 96tpi drives to transfer data to and from 48tpi formats such as Wren and PC/MS-DOS; we have never experienced the slightest problem.

4.3 Sector Identification

Each sector on a track has a number assigned to it - normally, the first sector on a track has number 0 or 1 (depending on the preference of the manufacturer of the computer the disc is used in), and the rest of the sectors generally number consecutively from this. Thus we can identify a sector uniquely by specifying which track it is on, and which sector number it is within that track.

For reasons of their own, Amstrad designed the disc formats used by the CPC range of computers to have a first sector number of 65 ($41) or 193 ($C1) on each track for system and data format discs respectively. Their CF2 and CF2DD formats for the PCWs adopt the more normal practice of starting their sector numbering at 1.

Even stranger systems, which the computer cannot naturally handle, are sometimes used by commercial software distributors to physically copy protect discs. Such techniques often involve wildly different sector numbering schemes on different tracks, extra tracks which CP/M cannot recognise, or sectors within particular tracks which have deliberately had peculiar faults inserted into them. Others format some tracks with sectors of different sizes. Such discs cannot then be copied by any normal CP/M copy program.

Similar problems can sometimes occur with discs formatted on other computers for use on the Amstrad. Sometimes the Amstrad's Disc Controller chip is not able to read the physical format of part or all of a track on a disc from another computer. Often, however, you will find that both computers **can** happily use discs formatted on the Amstrad using MFU. One good example of this is the Wren - discs used to transfer data between one of these and an Amstrad **must** be formatted on the Amstrad by MFU, **not** by the Wren.

Each sector is tagged by the controller with a **Data Address Mark (DAM)** to help it locate and verify the data the sector holds. Exceptionally, a **Deleted DAM (DDAM)** may be used instead. This need **not** mean that the data in the sector is part of a deleted file, although that was the original meaning and function of the DDAM. This dates from the prehistory of computing; nowadays, most computers use only normal DAMs.

If a sector with a DDAM **is** present on a disc, the Amstrad will normally ignore it and fail to read it. MFU's **A**)nalysis function **can** however detect, read and display a sector with a DDAM. If wished, it can then be written back out with a normal DAM and other functions and programs will then be able to read it.

4.4 Data Capacities

Obviously, the maximum amount of data which we can store on a disc is limited by the number of tracks it has, the number of sectors per track, and the size of each of those sectors. For example, each side of a standard CF2 3" format disc as used in Drive A of a PCW8256 normally has the following physical format :

40 tracks
9 sectors per track
512 bytes per sector

Thus such a disc can hold 40 * 9 * 512 = 180K (KiloBytes) of data. Note that this may be a slightly higher figure than the **actual** maximum free data space on a disc, for reasons which we will go into in the next section!

Other computers use many different physical formats - the number of tracks on a disc can be 35, 40, 77 or 80; the number of sectors per track varies from 4 to 26 or more, and the size of sectors can be 128, 256, 512, 1024 or 2048 bytes! This is one major reason for the apparent total incompatibility of discs from different computers - until MFU was written!

Other factors which affect the capacity of a disc are the **number of sides** which are used to store data (either one or two), and the **density** of the recording. Amstrad 3" CF2 discs as used in Drive A of the PCW8256 (and on the CPC464/664/6128) are rather unusual in that they are double sided, but only one side can be used at a time. This is because the drive which reads them only has a single read/write head. 5¼" discs are **not** like this - single sided discs **cannot** be "turned over" like a 3" disc, while double sided ones are used in drives with two read/write heads and so both sides can be accessed without the disc being removed.

The **density** of a disc is a description of the actual method used to record information on the magnetic coating. Early **Single Density** discs used the **Frequency Modulation (FM)** system, very similar to that used to record music on audio tapes. More modern **Double Density** discs use the **Modified FM (MFM)** system; as its name suggests, this technique packs information onto the disc at double the density of the older FM system, thus allowing twice as much data to be stored. All modern computers use Double Density discs; however, some older Single Density formats (such as Osborne and Acorn BBC) are still in fairly common use.

The PCW8256/8512 and CPC6128 **cannot** normally use Single Density formats due to hardware restrictions in Amstrad's Disc Controller circuitry. However, Appendix F details a simple series of hardware modifications to the PCW8256/8512's main circuit board which **will** then allow these computers to use both single and double density discs. Unfortunately, there are **no** equivalent simple modifications for the CPC6128.

Expanded CPC464 owners **can** use Single Density **without** any hardware modification.

We don't know whether CPC664s can handle Single Density or not, simply because we haven't been able to find one to test! If you own one of these rare machines, please contact us!

MFU is of course capable of setting up and using formats of **either** density.

4.5 Sector Skew

When a formatting program creates tracks and sectors on a disc initially, it may **not** number consecutive sectors within a track with consecutive numbers! For example, a disc with nine sectors, numbered 1 to 9, might actually have these written on each track in the order

1, 6, 2, 7, 3, 8, 4, 9, 5

It is important to realise that the **physical** order of sectors on a track has absolutely **no** effect on the order that data is read from the disc - it is **consecutively numbered** sectors which are read consecutively by CP/M, **not** consecutively **positioned** ones. So we can scramble the physical positions of sectors **without** scrambling the data they contain.

The reason for doing so is simple - it speeds up access to the disc. If you imagine CP/M (or a program) reading consecutively **numbered** sectors one after the other, then obviously each sector is read from the disc by the Disc Controller chip, transferred back to CP/M (which takes time) and then moved from CP/M's transfer buffer to its proper

destination in memory (which also takes time).

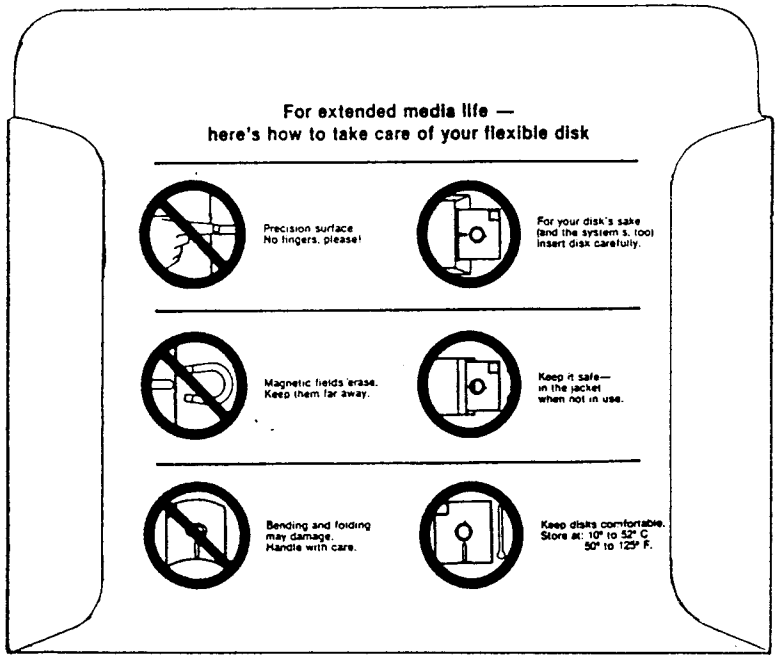
By the time CP/M asks the Disc Controller for the next sector, the next **contiguous** sector on disc will have slipped past the heads and (if it were also the next numerically), we would have to wait a full revolution of the disc (1/300th of a second, an age in computer terms!) for it to come round again. However, if we arrange the sectors physically in the order shown in the above example, then the sector which we have just missed is not the next one we want - instead, as if by magic, ours is the very next one to come along!

This system is known as **Physical Sector Skew**, and is implemented "once and for all" on any particular disc when that disc is formatted. If you want to change the **skew factor** (i.e. the number of sectors which physically come between two consecutively numbered ones) at a later date, you can only do this by reformatting the disc. Despite this restriction, physical skew is popular and widely used because it is simple to implement, effective and invisible to any operating system or program using the disc to read or write data. The alternative (and more complex) method of implementing skew - Logical Skew - is discussed in the next section.

Note that one of the features of physical skew is that different discs may be formatted with different skew factors and still used interchangeably - some will simply be more efficient (i.e. faster) in use; all will work perfectly.

4.6 Disc corruption and damage

Discs can be corrupted quite easily if handled wrongly, often in ignorance. Most disc sleeves have a list of "DOs" and "DON'Ts" printed on their backs; a typical example is shown below. You should make sure that these rules are observed by anyone handling your discs. Most of the ways you can damage a disc are physical - someone spills coffee over it, or it warps under the heat from the sun if you leave it beside a window.



It is also possible, however, to corrupt or even destroy the information stored on a disc without damaging the disc itself. This can happen if you put a disc too near a source of strong electromagnetic radiation or magnetic field, such as a television, electric motor, microwave oven or even an old telephone with a bell!

If a disc has been corrupted in this way, the only thing to do is attempt to recover as much information from it as you can (by copying any files which are undamaged to another disc) and then reformat the disc.

Appendix E details the way the Disc Controller reports the various types of disc errors, as well as offering suggestions as to the likely causes and possible remedies.

5. INTRODUCTION TO CP/M

This section explains a bit about the CP/M Plus operating system as used on the PCW8256/8512 and CPC464/664/6128, and describes the way it uses a disc to store information. Again, if you are familiar with this, you may wish to go on to the next section.

It is not possible to describe CP/M in any depth in a manual of this sort; the number of large books and long magazine series which have appeared on the subject over the past year are evidence of this! We can, however, explain the aspects which relate to disc formats.

CP/M is simply a very specialised program which is used to interact with, and control, the computer at the lowest level. When the computer is switched on, the files on disc which make up CP/M are read into memory and stored away. Although we don't know it, CP/M is **always** present in our computer's memory, though it may not always be active.

The first interaction we have with CP/M is when it displays the well-known

A>

prompt on our screens. When we see this, it means that CP/M is ready to receive a command. Broadly speaking, such a command can be one of two types - **resident** or **external.** A **resident** command is one which is "built in" to CP/M, such as **DIR. External** commands are simply the names of programs stored on disc - when CP/M gets such a command, it loads the appropriate program from disc into memory and **executes** it. This simply means that control of the computer is turned over to the program - CP/M is no longer in full charge.

When the program has finished, it in turn gives control back to CP/M so that we can enter another command. When a program **crashes**, this process does not occur, and we are left with a computer which will not listen to any commands we give it! In this situation CP/M has lost control completely, and our only course of action is to reset the computer to restart CP/M from scratch. Sometimes even this will not work and we have to turn power off to the machine to "kill off" a rogue program!

CP/M itself is split into two main parts - the **Basic Input/Output System** (**BIOS**) and the **Basic Disc Operating System** (**BDOS**). We never need to bother with this unless we are writing CP/M programs, but it will help us understand the use of discs if we explain the distinctions between the two parts.

The BIOS is responsible for directly controlling the computer's hardware. It is the BIOS that scans the keyboard for input, sends data to the CRT Controller chip for display on the screen, and communicates with the Floppy Disc Controller (FDC) chip to read and write the tracks and sectors of data on the discs.

The BDOS, on the other hand, is above all this! It is the BDOS which programs use to communicate with the outside world - a program simply asks the BDOS to read a file from the disc, for example; it is left to the BDOS to work out where on the disc the file is physically

stored. It then asks the BIOS to recover the file's data from disc by sending the FDC the appropriate commands.

The third part of CP/M - called the **Console Command Program**, or CCP, is the command interpreter which we actually talk to. The CCP displays the

A>

prompt on our screens, and is responsible for making sense of what we type in! If we give the CCP an **internal** command such as DIR, it will carry out the operation itself and display the results (a directory listing of files on our disc, in this case). To do this, it will call the BDOS interface routines in exactly the same way as any other program. In fact, you can consider the CCP to actually be a program just like any other; the only real difference is that it is always hidden away somewhere in the computer's memory, ready to run. It does **not** need to be loaded from disc each time it is needed, as other programs must be.

The CCP is an example of a type of program called a **shell.** The concept of shells is not used very much in CP/M, unlike other operating systems such as PC/MS-DOS and Unix, where it has a much greater significance. It **is** possible to replace the standard CCP with a shell which you have written yourself, if you have a need for unusual commands or facilities, but this requires an expert knowledge of CP/M, and is not for the timid!

Although the division of labour between BIOS, BDOS and CCP may seem needlessly complex at first sight, it is in fact this which makes CP/M as useful and widespread as it is! The BDOS provides a standard interface for programs to use to perform all their input and output (whether it be to screen, printer, discs, keyboard or any other device attached to the computer). This interface is the **same** on **all** CP/M computers, of whatever make - thus a program which runs on one CP/M computer should run with little or no alteration on any other!

The BDOS in turn has a standard interface to any individual computer's BIOS; all a manufacturer has to do is write the code for the BIOS part of CP/M, which has to know about each computer's hardware design. Once this has been done, the rest of CP/M and any properly-written programs will run on that computer.

The CCP, in its turn, has the function of providing a completely standard "User Interface" - if you know how to use CP/M on one type of computer, you will feel equally at home on any other!

5.1 File Structure

CP/M (in common with all other computer operating systems) has the function of imposing and maintaining a **file structure** on a disc which physically is simply a vast and uncharted ocean of data! We (and our programs) can then store our data on a disc by putting it into **files** which each have unique names and so can be recalled 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, CP/M lumps sectors together into **groups**, and allocates storage on the disc to these groups. It identifies data groups by number, and can calculate the track and sector position of the start of any group by using some of the values which are held in the **Disc Parameter Block** (DPB).

The DPB is a block of data maintained by the CP/M BIOS for use both by the rest of CP/M and by programs. It is not necessary to go into more detail about it here; Appendix D contains a full description of all the Amstrad's disc parameters.

CP/M stores the numbers of groups allocated to a particular **file** in a **directory** which it maintains at the start of the disc. A file is simply a collection of data groups which we have asked CP/M to keep track of for us when we created it - for example, it could be a program or a wordprocessed letter; CP/M neither knows its content nor cares. The directory is simply an indexed list of files stored on the disc which allows us to recover them and store others.

CP/M's group numbers are stored and used **entirely** internally to itself - we never need to know about them other than to understand the way information is stored on disc. Each directory entry contains the name we have given a file, information as to whether it has any special attributes such as Read-Only or System, and (most importantly) a list of group numbers which, in order, hold our file. One point to note is that the groups holding a file do **not** need to be contiguous - as CP/M hold a record of **every** group used by a file, it can allocate space 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 from CP/M on a disc which may only have half the theoretical maximum number of files stored on it. The way CP/M allocates directory entries to files is complex, and we'll leave a fuller description to Appendices C and D.

5.2 Logical Disc Structure

CP/M discs may also have **Reserved Tracks**. These are tracks on the disc which are not available to us for data storage, and so discs with reserved tracks have a lower capacity than similar discs without. Reserved tracks are used to store the programs which make up part or all of CP/M itself; this code is read into memory from the disc and executed automatically when CP/M is booted.

In a banked-memory CP/M Plus system such as is implemented on Amstrads, only the disc which is booted from at the start of day need have reserved tracks; however to avoid confusion Amstrad decided that **all** discs used on PCWs would have one reserved track. CF2 and CF2DD discs therefore have a slightly lower capacity than is strictly necessary. On CPCs, SYSTEM format discs have a reserved track and thus a slightly lower capacity than DATA format discs.

Amstrad have made use of some of the wasted space on CF2 and CF2DD format discs to implement their "smart" auto-format detection system which gives PCW8512 owners a kind of limited multi-format ability - their otherwise incompatible Drive B can read but not write discs from Drive A!

The directory, group size and file structure that CP/M imposes on a disc is known as that disc's **Logical Format**. It is important to distinguish this form the disc's **Physical Format** - the logical format controls how and where files are stored and recovered, but is wholly a function of software; it does not (and cannot) affect the way data is actually recorded on the disc in physical tracks and sectors.

Nowhere does this cause more confusion than in the two quite different ways CP/M uses the word SECTOR! Early CP/M computers used discs whose physical sectors were 128 bytes in size, and so CP/M itself was designed to use 128 byte sectors in its interface with programs. However, times moved on and CP/M didn't!

Computers such as our Amstrads have much higher-capacity discs than the earlier types and so generally use physical sectors which are much larger than 128 bytes (512 bytes per sector, in our case). However, CP/M programs still "expect" to read and write 128 byte sectors when they call the BDOS Disc Service Routines.

To avoid confusion, we will always refer to one of CP/M's **logical**, 128 byte "sectors" as a **RECORD**, and leave the term **SECTOR** to refer only to the **physical** sector size used by a particular disc format.

5.3 Logical Sector Skew

One fairly complex aspect of CP/M which requires some discussion is the concept of **logical sector skew.** In the previous section, we looked at the idea of **physical sector skew** as a technique for speeding up the rate of data flow to and from a disc. This system optimises the transfer and processing timing by staggering the physical positions of sectors on a track, interleaving them in a numerical pattern so that consecutively numbered sectors are not necessarily physically adjacent.

Logical skew, on the other hand, makes sector interleaving **entirely** the responsibility of software - the BDOS, in fact. With physical skew, we didn't have to worry about where sectors were after a disc had been formatted, as the sector pattern was laid down at that time. Logical skew, however, requires a disc on which the sectors are optimally formatted in simple, non-interleaved numerical sequence and all interleaving is performed by software when data is read or written.

Where a format has logical skewing, the BIOS maintains a **Sector Translation Table.** This is a list of interleaved sector numbers which looks just like the actual order of sectors on an equivalent physically-skewed disc. Whenever a sector is to be read or written, the BDOS first refers to this table, using the **logical** sector number it has been asked to access to index into the table and extracts the **physical** sector number where the data is located. It is **this** sector number that is then passed to the BIOS for action.

This means that when you look at physical sector 1 on a logically skewed disc, you won't find the next, contiguous piece of data in physical sector 2! This makes determining the format for such a disc quite difficult. MFU's **A**)nalysis option will give the **physical** format, and display numerically consecutive physical sectors; however, you will still not know the logical pattern used for data access unless you are lucky enough to have the appropriate technical information for that format. You may not even be able to locate the different parts of the directory!

To determine an unknown logical skew pattern, you will have to experiment by setting up different logical skew factors or patterns using MFU's **E**)dit function and trying to read a large text file from the mystery disc until it all displays without spurious sectors popping up in the middle of it! This problem is discussed further in Section 6.2 (**A**)nalyse unknown disc).

6. USERS' GUIDE TO MFU FUNCTIONS

MFU's Main Menu contains the following options:

A)nalyse unknown format
C)opy files between discs in Drive B
D)isplay Drive B format
E)dit Drive B format
F)ormat disc in Drive B
L)ibrary manager
N)on-CP/M file transfer
P)Produce stand-alone format setup Program
S)et system defaults
T)rack-by-Track backup of a disc in Drive B
V)erify disc in Drive B
EXIT to CP/M

In addition to these, ^P entered at **ANY** prompt within the system will cause a screen dump to be sent to the printer if it is attached. This can either be in normal draft mode, or High Quality.

At **any** time, of course, you can use a PCW's own native screen dump facility by pressing **Extra** + **Ptr**, but this will produce a much smaller screen image on the paper than MFU's screen dump.

Each of the above options, and their sub-menus, will now be described individually.

6.1 A)nalyse unknown disc

This option gives you access to one of MFU's most powerful modules. The Disc Analysis software which has been integrated into Version 4 of MFU has the capability to determine **all** of the physical parameters affecting disc format, including disc density, number of sides and track pitch. **Any** track can then be analysed to determine the number of sectors it contains, their numbers and physical order within the track (to determine the physical skew pattern).

In addition to being **the** most powerful disc analysis system around, you also now have the all the power of a full-feature disc sector editor available - the sort of thing other people sell separately for more than the cost of the whole of MFU! You can read and browse through any sector on disc, displayed in both hex and ASCII, make any changes to the data you want and write the altered sector back to disc.

All these facilities are extremely useful if you have an unlabeled disc and want a clue as to what format it is, or a disc you thought was formatted produces disc errors in use.

In the former case, you can use the full analysis functions to determine the physical format, then use the sector editor to browse around the first few tracks, finding out where the directory lives, whether or not it appears to be contiguous (if not, this suggests logical skew is present), and inspecting the Group Allocation information in a directory entry for hints as to the blocking size used.

In the latter case, you can analyse the track which appeared to be damaged and check the suspect sectors for yourself.

When the disc and its tracks are analysed, MFU automatically configures the sector editor so that you are always prompted with displays of valid ranges for head, track and sector numbers to select from.

We'll be assuming in this section that you are familiar with the concepts and terminology of physical disc formats; if you are a bit shaky on this, we suggest you go back a bit and read Section 4 before proceeding any further.

6.1.1 Opening Menu

When you first enter the analysis module, you will see the main Analysis Opening Menu, though it will show very little information at this stage. At the bottom of the screen you will see the options which are available; these are listed more or less in the order in which you would use them.

We'll explain what each does in turn by describing how you would analyse a disc and edit some of the data in one of its sectors.

6.1.2 A)nalyse Disc Format

The only option which is available to us right at the start is this one - after all, MFU can't be expected to tell us things about a disc it hasn't looked at yet! When we select this option by typing **A**, MFU carries out a fairly complex series of operations to establish the overall physical format of the disc currently in Drive B.

When the analysis process has been completed, MFU displays the results on the top line of the screen. For example,

Disc is Double Sided, Double Density, 96 tpi (80-track).

which tells us how many sides the disc used (either just one or both), the **density** of the data (either single or double) and the **track pitch** (either 96 tracks per inch or 48 tpi). The first of the two pitches is used by 80-track discs; the other by 40 or 35 track discs. Please note that the **(80-track)** displayed here by MFU is merely a guide - there is no guarantee that there actually **are** exactly 80 tracks on the disc! Some peculiar formats only use 77 of them, while others sneakily use 81, usually for copy protection against software piracy.

We can now either **A**)nalyse another disc, or move on to analyse a particular track. To do this, we must first tell MFU which **Head** we want to use on the disc drive (if the disc is double sided), then which track to move that head to. We do this as follows.

6.1.3 H)ead no.

Select this option from the Opening Menu. If the disc was found to be double sided when analysed, MFU will ask you to choose the head to use (i.e. which side of the disc do you want to look at!). If the disc is single sided, then you are forced to use Head 0.

6.1.4 T)rack no.

Once the head has been selected, use this option to select the track to be analysed. You will be prompted for a valid track number within the physical range permitted by the track pitch which MFU detected while analysing. Note that you are actually allowed to look at the 81st (or 41st, if the disc is 48tpi), track - all disc drives can physically access the extra track, and as we mentioned earlier, some copy-protected software uses it.

When you have entered a valid Track number, MFU will analyse that track. The results are shown at the top of the screen, in a display such as

```
Disc is Double Sided, Double Density, 96 tpi (80-track).
Head 0, Track 0 has 9 sectors, each of 512 (&0200) bytes.
Physical Sector Sequence :
Decimal :    1  6  2  7  3  8  4  9  5
Hex     :    1  6  2  7  3  8  4  9  5
Probable gap sizes :  Gap3(Format) = 90     Gap3 (R/W) = 46
```

If the inter-sector gaps are smaller than MFU can reliably analyse, the additional message

WARNING - Gap3 too small for accuracy. Sector Table may NOT be accurate.

will be displayed. This does **NOT** mean that there is anything wrong with the disc - it is just that MFU cannot guarantee that **the order** of the sectors shown is right. All the other information (such as number of sectors, their numbers and sizes) **is** still reliable.

A more serious problem is present if instead you see the message

ERROR - Track format is INVALID.

This means that when MFU calculated the total space required on the track to hold the sectors, header information and gaps, there wasn't enough room! This can occur for two possible reasons.

Firstly, with discs which are formatted in a non-standard way, with unusual values in the Sector Headers. You will get this problem with some "protected" software distribution discs and one or two "enhanced" CP/M formats which are designed to be awkward to simulate. You may also see the Sector Size shown as a "?", indicating that MFU could not determine the correct value from the gibberish in the header.

The misleading information on such discs is designed to confuse the Disc Controller; to access data on such discs you will in general need complex and specific software which can get round these problems. MFU will probably fail to read such discs reliably; standard CP/M programs will definitely fail!

The second way this error can be generated is where the drive speed tolerance required by your drive is greater than that for the system from which the disc came. It may be that the other computer has taken advantage of superior performance to squeeze more data onto a track than MFU thinks your drive should be able to access!

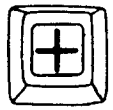
MFU will now automatically attempt to read the first physical sector on the track into its data buffer and will display the contents on the screen ready for examination or alteration by the sector editor. If the track format is invalid, this operation may fail and the FDC error codes will be displayed.

We can use the sector editor to look at the actual data held in any of the other sectors on this track, by selecting the next option.

6.1.5 S)ector no.

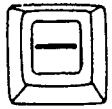
Here you will be presented with a choice of sectors to look at; again, the range is restricted to the sector numbers MFU detected when it analysed the track.

When you have entered a valid sector number, that sector will be read from the disc and the first 128 bytes displayed on the screen. If the sector is larger than 128 bytes, you can look at the **next 128 bytes** of the sector by pressing the



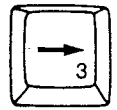
key.

To go backwards again and look at the **previous 128 bytes** of the sector, press the



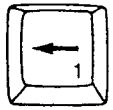
key.

If you want to look at the **next sector** on the track, press the



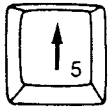
key.

This will read the next sector from disc and display its first 128 bytes on the screen as before. Similarly, if you want a look at the **previous sector,** press the



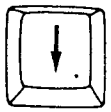
key. If you want to select another sector number explicitly, just press S again.

To select a different track, just press T as before. However, you can step to the **next track** simply by pressing the



key.

Pressing the



key will step you to the **previous track.**

If you suspect that the meaningless garbage you are looking at on the screen might be sensible but **inverted** data, the "*" key will toggle between normal and inverted data display.

If MFU detects that the sector is tagged on disc with a **Deleted Data Address Mark** (DDAM), this fact will be displayed on the top line of the display, to the right of the Track, Head and Sector numbers. If you are not familiar with concept of Data Address Marks, don't worry! It's a technical distinction used by the FDC when reading and writing data; a DDAM need **not** mean that the sector has actually been deleted in any way. A fuller discussion of the subject can be found in Section 4.

At any time, we can edit a sector by selecting the next option.

6.1.6 E)dit sector

If you select this option, you will notice that the bottom area of the screen clears, and the normal Opening Menu is replaced by the Edit Menu. In addition, the cursor is positioned over the first byte in the display of the sector's data.

If you type in a new hex value, it will overwrite the previous value at that position. You'll also see the ASCII character represented by the byte change.

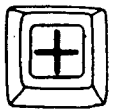
Normally, the character corresponding to the full 8-bit value of the byte is displayed in the ASCII field. However, if you want to restrict this to a display of standard 7-bit ASCII, ignoring the top bit, then pressing the "#" key will do this. This key is a toggle - press it again and the display will switch back to 8-bit mode.

If you press the TAB key, the cursor will be moved from the hex display field over onto the equivalent byte in the ASCII display field at the right of the screen. You can now enter new values as characters rather than hex codes. Note that only normal ASCII characters in the range &20 to &126 can be entered in this way, regardless of display mode. ASCII entry mode is extremely useful if you want to change **text** rather than code or data.

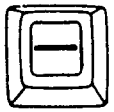
The TAB key is a toggle - pressing it again will return you to the hex field of the display.

Pressing the **SPACE BAR** moves the cursor to the next byte; pressing **RETURN** will move you down to the start of the next line. The **CURSOR KEYS** will move you around the part of the sector shown on the screen. All the movement keys "wrap round" (top-bottom, left-right) on the display.

To edit another part of the same sector if it is larger than 128 bytes, use the



and



keys as before.

If you are displaying data in inverted mode, then MFU will automatically invert all your changes internally before you write the sector back to disc.

When you have made all the changes you want, press **EXIT** to return to the Opening Menu.

6.1.7 W)riting and R)eading sectors

Note that NONE of the changes you have made have yet been made to the disc. Only the copy of the data in MFU's memory buffer has been altered.

If you want to permanently record your changes on disc, use the **W** option to **W)**rite the data. If you want to scrap all your changes, you can use the **R)**ead sector option to re-read the original version of the sector from disc, overwriting your changes. You could of course edit the data again to make more changes before saving it.

When you select the **W)**rite sector option, you will be offered a choice of sector Data Address Marks. The default will be the same as was detected when the sector was read - you can select this by typing "Y" or <RETURN>. Typing "N" will write the opposite type of DAM to the sector header. For an explanation of DAMs please refer to Section 4; if you don't know anything about them then it is safe to just type <RETURN> to accept the default - changing the DAM is only done in exceptional circumstances.

For explanations of the various Disc Error Codes which MFU reports, please refer to Appendix E.

To exit the analysis module, press **EXIT** from the Opening Menu and you will return to MFU's Main Menu.

6.1.8 Analysing an unknown disc format - example.

As an example, we'll describe how you would go about analysing a disc whose format is unknown using MFU's Analysis module to determine physical format and estimate logical structure.

Let's look at a disc which is actually Amstrad CF2DD format (though we don't know that officially!). You can work your way through this example by **A**)nalysing any of your normal CF2DD discs.

Firstly, let's get the exact **physical** format. Put the disc into Drive B and run MFU. Type **A** to enter the Analysis module. Now type **A** again to analyse the overall disc format. MFU will examine the disc and report on its density, number of sides and number of tracks per side. In our case (with a CF2DD disc), these will be **Double Density, Double Sided** and **96tpi (80-track)** respectively.

Now let's have a closer look at Track 0 - the very first track on the disc, and so usually a very good place to start! Type **H** and set the Head number to 0; then type **T** and set the Track number to 0 also. MFU will now read and analyse that track, and report the number of sectors on it, their numbers, sizes and probable gap spacings. Select **H**)ead 1 and check that the format of Side 1 of cylinder 0 is the same as Side 0 - if the sectors have different numbers, you probably have Sector Wraparound Cylinder access on your hands. Fortunately, in this case we don't!

We now have the **physical** format of the disc entirely specified. In the case of our "mystery" disc, we have a Double-Sided, Double-Density disc with 80 tracks per side; each track has 9 sectors, each of 512 bytes, numbered 1 to 9.

Now to try and find the **logical** format. This is the way the data is organised on the disc, and MFU can't offer very many suggestions about this.

If you suspect that your disc might be a MS/PC-DOS variety, then it is a good idea to try it first using MFU's **N**)on-CPM option rather than low-level analysis. The **I**)BM transfer module itself performs some very sophisticated analysis to determine whether a disc is in DOS format, and this will tell you a lot faster and easier than looking through the disc sectors yourself! Similarly, if you think a disc is probably from a BBC, try it with the **B**)BC transfer sub-option before analysing it yourself - again, this module will perform a more intelligent analysis of the disc and report if it seems to be a BBC format.

If, however, both of these have failed (or you reckon it's a CP/M disc in the first place), then the following steps are recommended. Please note that to proceed much further, you really require a reasonable understanding of CP/M directory structure and disc parameters. These are covered in Appendices C and D.

1. Try to determine the number of reserved tracks and the sidedness by searching for the start of the directory. If you are not familiar with the appearance of directory entries on disc, please refer to Appendix C, where directory structure is described in detail.

2. Determine the size of the directory by searching for the start of the first file's data, watching for the possibility of logical skew.

3. Use the information in the directory, along with your knowledge of the disc's physical structure and capacity, to estimate the likely Block size.

4. Exit **A**)nalysis and use **E**)dit to construct an appropriate format in memory. Then exit MFU and try to **DIR** the disc; if files are present try to **TYPE** some text or run a program.

5. Go back to MFU and try again when it doesn't work quite right....!

Right; lets start!

1. Reserved Tracks.

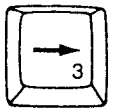
After we have Analysed Head 0, Track 0 we can use the **S** command to choose a sector on the track for examination. Select Sector 1 (the first one on the track). MFU will read it in and display the first 128 bytes on the screen.

It is obvious that whatever this is, it isn't the directory! In fact, if we were clever, we could save ourselves a lot of work by realising that it is in fact the Format Descriptor Block, and contains everything we want to know! However, we'll pretend we didn't see it, as only Amstrad discs have this, and you won't normally be analysing Amstrad discs! Appendix D describes this block if you want to know more about it.

Right; Track 0 on Side 0 looks to be reserved. Try Head 1, Track 0 and look at the first sector on the other side. What do you know - directory! So now we know that the disc has one reserved track, and is accessed as **Double Sided Standard Cylinder.** (We know the latter because if either of the Side Access techniques were used, the first track on Side 1 would contain data or unformatted pattern (&E5), **not** directory - it would all be on the **first** side (0).).

2. Directory size.

We have already read the first sector on the track into memory; now use the



key to get the next **physical** sector on the track. Check that this either has directory entries or is filled with formatted pattern (&E5). Keep on reading consecutive sectors until either you suddenly find one which is filled with data or you run out of track!

You will be able to spot the transition to data very easily. Directory entries have a nice, regular structure which you can usually spot a mile away, and in any case, few directories have ever been completely filled so normally you'll get a few sectors of unused &E5 formatting pattern. Data is obvious by its total randomness! If you're lucky, the first file on the disc will be text and you'll be able to read parts of it in the ASCII dump at the right side of the screen.

Once you have determined the total number of physical sectors which are used to hold the directory, you can work out the number of directory entries. There are 4 directory entries in each 128-byte directory record, so each of our 512-byte physical sectors will hold 4 x 4, or 16, directory entries. On a CF2DD disc, you'll find that **all** of Head 1, Track 0 and Sectors 1 through 7 of the next track (Head 0, Track 1) are reserved for the directory; this gives **16** physical sectors in all (remember that there are 9 per track) and so the directory takes up **8K.**

Fortunately, on a CF2DD disc we **don't** have any logical skew to contend with. If this **is** present, it makes life difficult as we have no way of finding out what the skew factor is other than by using our eyes and guessing! You can spot logical skew by the mess it makes of the directory when we look at it in physical sector order with Analyse. Sectors of directory information are interleaved with either the end of the directory (sectors of unused &E5 pattern) or file data from the first file.

To sort it out, you have to try and find the physical sector which appears to contain the second physical directory sector. The easiest way of doing this is to note the last Block allocated in the last directory entry in the first sector, and then search for the sector whose first directory entry allocates the next consecutive block to a file. The number of physical sectors which were between these two adjacent parts of the directory gives the Logical Skew Factor which you can in turn enter into the format you set up.

3. Block size.

Look at the start of the directory and find the lowest block value allocated to a file. This will normally just be the first block number of the first file's entry, but it need not be - check it actually **is** the lowest before you jump to conclusions!

We can now calculate the size of the disc's blocks quite simply. The directory occupies the blocks from 0 to (First File Block - 1); 4 blocks, numbers 0 to 3 in our example; and we already saw that the directory took up 8K in total. So the block size on the disc is 8 / 4 = 2K!

4. Try it!

We have now derived all the format information for the disc (we think correctly!). Exit the analysis module and use the **E**)dit facility to enter the format. We want to set up the following :

```
Sidedness? (0..4)              1
Tracks per Side?               80
Sectors per Track?             9
Sector Size?                   512
Density?                       1
First Sector?                  1
Reserved Tracks?               1
Block Size?                    2K
Directory Entries?             256
Inverted Data?                 0 (we could read it!)
Auto Format detect?            0 (we found an AF block)
Physical Sector Skew factor =  2 (don't know this, but
                                  2 is a good start!
```

As we enter these, MFU will automatically calculate the optimum Gap sizes for us. Physical skew is worth putting in as 2 for a CP/M format if you don't think it has any logical skew; it might not be optimal, but should give a reasonable performance. You don't need to worry about it anyway unless you plan to **format** more of these discs.

Although the example of CF2DD we chose here is useful because it is straightforward to analyse and familiar, it has the drawback that it is the default Amstrad format and normally its parameters will already be correctly setup and there will be nothing for us to change!

You will get a better sense of setting up a new format if you load something obscure from the library before you analyse the CF2DD disc - when you edit the parameters, things will change and at the end (with any luck!) you'll see MFU identify the format as CF2DD, confirming your diagnosis!

6.1.9 Problems and hints!

Unfortunately, life isn't always as straightforward! Many CP/M formats can look **extremely** confusing to analyse, and all are unique, usually with unique problems! And if you are analysing something which isn't even CP/M, then good luck to you!

The more information you have to start with, the less you have to extract from **A**)nalyse and the easier it is to spot when you haven't quite got it right! Get hold of the other computer's technical manual and scour it for any and all clues or mentions it makes of its disc format. On a CP/M Plus computer, use the command

B>**show b: [drive]**

to get a listing of the disc's CP/M format. This will tell you the number of directory entries, block size and all the rest. You'll even get the physical sector size!

On a CP/M 2.2 computer, the equivalent command is

B>**stat dsk:**

This will return the same information as above, but **won't** tell you the physical sector size - you'll have to use Analyse for that. Once you know the physical sector size, check that CP/M's number of records per track (the **Sectors Per Track** figure from the **show** display) corresponds with that which you can calculate from the number of physical sectors and their size which Analyse tells you.

If they are **not** the same, then the probable explanation is that your Amstrad can't read the very first sector on each track due to a peculiarity in the way the FDC chip works. This means that Analyse won't know that it is there and will therefore report one sector too few on the track format display, and happens with some formats (e.g. Wren) from computers which use a different type of FDC.

Often you can get round this problem by setting up a physical format as described by your Analysis, but with an extra first sector with a number one less than the first displayed by Analyse. If you then format all discs for use on both machines **on your Amstrad** using this format, you will usually find that **both** computers are quite happy to use them.

Once you have set up the format on the Amstrad using MFU, exit to CP/M and use the **show** command to see if your Amstrad's CP/M really thinks the format **is** the same as the original.

Even all this information, though, can't tell you the access method, whether skew is used (and whether it is physical or logical!), or whether data is inverted, for example. Again, you'll still have to use Analyse to determine this sort of thing.

As we mentioned earlier, Logical Skew can usually be spotted by the "gaps" it appears to make in the directory or a text file whose contents you are familiar with. However, some formats don't use a regular and identifiable skew **factor** at all, but instead have a defined **sequence** of logical sector numbers. This could be anything, and you'll just have to

try out all the permutations you can think of! Usually, it'll use the one which you **didn't** think of...

Side Access formats can be tricky to get right - you can usually spot that they aren't Cylinder Access from the location and layout of the directory (it'll be on consecutive tracks on Side 0, with data or format pattern on the other side) but deciding whether it is Track Order or Wraparound can be tricky! In general, pick one and suck it and see!

Some formats (e.g. Epson QX-10) use a fiendish system of **Logical Sector Wraparound**, where it considers and entire cylinder (i.e. **both** sides) to be a single, contiguous track and therefore "thinks" it has half the total number of tracks but twice the number of sectors on each! This can usually be fiddled quite simply by setting Cylinder Access and allowing **twice** the number of **reserved tracks** to what Epson (or whoever) tell you it has! Examine the Epson format in the library and try to see how it works.

Such formats will either have access modes **1** or **4**. You'll have to determine which by looking at both sides of the disc using Analyse. If the sector numbers are the same on both sides then it's mode 1; if they are contiguous it's mode 4.

Some discs will not use all their available space, and hence you'll have to enter a lower number of Tracks per Side than the usual 40 or 80 (again, the QX-10 is unfortunately a good example of this!). Some older formats (such as early Superbrain) use only 35 tracks on a 40 track (48tpi) pitch, and some others emulate 8" disc formats by only using 77 tracks on an 80-track (96tpi) pitch.

6.2 C)opy files between discs in Drive B

This option allows you to transfer selected files between 5¼" CP/M discs in drive B. The discs can have different formats or they may both have the same format.

Copying of files between two discs is achieved by using the RAM disc as a buffer to temporarily store the files if you have one; otherwise, you will be prompted to insert a blank 3" disc into Drive A for use as a buffer. The selected files are first copied from the 5¼" source disc onto the buffer drive then, when you have changed discs, the files are transferred back out from the buffer onto the new destination disc.

Of course there may not be enough room on the buffer drive to hold all of the files, in which case the transfer will proceed in stages with as many files as possible being transferred each time. This will mean changing discs more than once during the transfer process but the number of changes can be kept to a minimum by ensuring that you have as much free space as possible on the buffer drive.

In order to use this option at all you must have at least 64K available on the buffer drive, since any less than this would involve too many disc changes. When you select this option you will be told how much space is available on the buffer drive and, if this is sufficient, you will be allowed to proceed.

Before you start you will be asked to select the formats of your source and destination discs. This is done by selecting the appropriate format number from the list of formats which will be displayed.

When you have done this and inserted the source disc you will be asked to enter a **filespec** for the files you wish to copy. This may be anything from a single filename, such as PROGRAM.COM, to a filename containing **wildcards**, such as ***.DOC** (this would specify **ALL** files on the disc which have the extension **.DOC** - see your PCW User Guide for a full explanation of wildcards in filenames). Simply pressing RETURN will allow you to specify all files on the disc. Similarly, you must enter the CP/M user number where your files are to be found.

The directory of the disc will then be scanned to find if there are any files which match your specification. If so then the number of matching files found will be displayed and you will be asked if you wish to copy them all. Answering "N" will cause a list of all matching filenames to be displayed and you may select which of these you wish to transfer.

After you have chosen the required files then they will be transferred, via the buffer, from the source disc to the destination disc. As each file is copied its name is echoed to the screen to keep you informed of the program's progress. As the copying proceeds you will be prompted to insert the **SOURCE** or **DESTINATION** disc at the appropriate times. (Note that it is good practice to write-protect the source disc in case you accidentally insert it in place of the destination disc.)

If all goes well you will be informed when the copying has been completed and you may then return to the Main Menu. If, however, you have insufficient room on the destination disc then the transfer will be aborted and an appropriate message will be displayed. If you wish to copy the remaining files then you must repeat the copying operation using a new destination disc.

6.3 D)isplay Drive B format

Selecting this option will produce a comprehensive listing of all the parameters contained in the XDPB (eXpanded Disc Parameter Block) for the disc format currently installed for Drive B. A detailed description of all these parameters is given in Appendix D; this section gives a more general explanation of only those which are particularly relevant to us here.

The first eleven entries (**SPT** to **PHM**) make up CP/M's Disc Parameter Block (or DPB), and are displayed for information only. It is not necessary to fully understand these parameters to understand or alter a format - they are required by the higher level functions of CP/M, but all are calculated internally by MFU when a format is loaded or defined.

The next nine make up the eXtended DPB. This is a custom addition to Amstrad's implementation of CP/M, and is used to specify physical parameters to the BIOS concerning the disc format - for example, the first sector number on each track and the order in which the tracks on the disc are accessed by the BIOS.

The first of these nine entries, **sidedness**, details the exact method which is used by the format to access data stored on the disc. Where the disc is **Single Sided**, there is no complexity - data is simply written to, and read from, consecutive tracks starting at 0. All tracks are on the same side of the disc. However, the trouble starts when discs are **Double Sided**! Different computer manufacturers have adopted different schemes for using the two sides.

Cylinder Access is the technique where each track on the bottom side of a disc (**Side 0**) is considered to form a **cylinder** with the corresponding track of the same number directly above it on the top side of the disc (**Side 1**). Data is then read from, and written to, the disc in the order :

Track 0, Side 0
Track 0, Side 1
Track 1, Side 0
Track 1, Side 1
Track 2, Side 0

and so on.

This is the system which is probably the most common - it is used by many different types of computers, from the PCW8512 to most 16-bit MS/PC-DOS machines such as the IBM-PC and compatibles such as Amstrad's PC-1512/1640.

Side Access is the alternative scheme to Cylinder Access - or, to be more exact, includes two similar (but wholly incompatible!) systems. Both of these systems read and write data to all the tracks on Side 0 in order (as if the disc was Single Sided); they then switch over to Side 1 and repeat the process.

However, the two Side Access schemes differ drastically in how they work once they start using Side 1! Many computers restore the heads to Track 0 and start working outwards on Side 1 in exactly the same way as

they did for Side 0. This is usually referred to as Conventional, Standard, or **Track Order** Side Access. Unfortunately, in their wisdom Amstrad decided not to support this system within their BIOS - so MFU has to alter the BIOS so that it **can** be used!

Instead, Amstrad support the alternative technique, often referred to as **Wrap-Round** Side Access. This system does **not** restore the heads and start using Side 1 in the same track order as Side 0. Instead, the heads work **back** along Side 1 in **reverse** track order - from 79 back to 0 on an 80 track disc, or 39 back to 0 on a 40 track disc.

A fifth (and rarer) variation is **Cylinder Access with Sector Wraround.** This is identical to the simple cylinder access technique discussed earlier, except that the sectors on each the two tracks forming a cylinder have **different** sector numbers. For example, the track on Side 0 might have ten sectors numbered 1 through 10; the track on Side 1 would then have ten sectors numbered 11 through 20!

Although this may not seem particularly complicated, it causes a lot of problems at BIOS level on computers such as Amstrads which were never intended to use such a technique. MFU (as always!) patches the BIOS to allow this access method to be used, but in this case some software may not be able to use a disc with this access method. You should test formats which use this method thoroughly with particular programs before trusting them fully.

MFU displays the five available alternatives as

Single Sided
Double Sided **Std. Cylinder**
Double Sided **Track Order Side**
Double Sided **Wrap Round Side**
and **Double Sided** **Sector Wrap Cyl**

respectively.

The second and third entries in the XDPB, **Tracks per Side**, and **Sectors per Track**, simply specify the number of tracks on each side of the disc (and thus the number of **cylinders** on a double-sided disc), and the number of physical sectors which are present on each of these tracks.

The **First Sector Number** entry following these specifies the number which identifies the first physical sector on the track. As we saw in Section 4, each sector on a track has a block of "hidden" data attached to it which contains information to allow the Disc Controller chip to locate the sector on the disc and check its validity. Part of this data is a **Sector ID** number. Normally, sector numbering on each track is the same as on every other track, and is consecutive, normally starting at either zero or one. For their own purposes, Amstrad designed the **System** and **Data** formats as used on the CPC464/664/6128 with unusual first sector numbers. These are 65 (&41) and 193 (&C1) respectively!

Sector Size simply defines the size, in bytes, of the sectors on the disc. This must be either **128, 256, 512, 1024** or **2048** bytes. **All** the sectors on a disc must normally be the same size.

The next two entries, **R/W** and **Format gap lengths**, are concerned with the way space is allocated on each track of the disc. The sectors on a track are **not** physically contiguous - there are gaps left between them. This is necessary to compensate for speed variations in the disc drive - if the sectors **were** contiguous, then when a sector was written to disc there would be a real risk that it would "miss" its place and overwrite part of the sector before or after it.

The detailed calculation of gap sizes is quite complex and is described fully in Appendix B; however, it is not necessary to understand this to define formats within MFU. As will be explained in the next section, MFU will automatically calculate the optimal gap sizes for your format if you wish.

Density determines the actual recording method which is used to write data on discs, and was described in more detail in Section 4. There are two types of Density - **Single** and **Double.** All modern discs use Double Density; some older Single Density formats are however still in use.

Auto-Format Detect is an Amstrad-specific parameter. It should usually be set **On** when drive B is being used in a standard Amstrad format (for example, as the normal CF2DD second drive), and **Off** for **all** non-Amstrad formats. If ON, it forces the BIOS to determine the format of any disc inserted into Drive B, and self-reconfigure to that format, automatically. It does this by using an **Auto-Format Descriptor Block** located in the first Sector of logical Track 0. Obviously, if a non-Amstrad format disc is put into drive B while the Auto-Detect flag is set on, then the BIOS will get very confused; it may in fact be impossible to recover from the errors this generates without resetting the computer.

Please note that the Auto-detect system supported by MFU is that used on the **PCW8256/8512, NOT** the earlier and more primitive system implemented on the CPC464/664/6128. IF you are using MFU on a CPC464/664/6128, you should **never** attempt to install a format which has the Auto-format detect flag set to ON other than for the purpose of formatting a disc for future use on a PCW8256/8512. After formatting, you **must** edit the format installed in memory and set the Auto-Detect flag to OFF **before** you make any other attempt to access Drive B.

The Auto-Detect system on CPCs works simply by using the sector numbering system to define the various formats; attempting to use **any** other format of disc in Drive B (**including** PCW formats) will result in the unusual error message

Drive B: Bad format - Retry,Ignore or Cancel?

being displayed by the BIOS, which on CPCs can lead to CP/M crashing without warning!

Consequently, **all** formats defined in MFU's library file have the Auto-Format Detect flag set to **OFF** on CPC versions, **including** PCW CF2 and CF2DD formats. This means that discs formatted normally on a CPC will **not** have the Format Descriptor Block written to Track 0, Sector 0.

All discs for use by a PCW should therefore **either** be formatted on the PCW, **or** have the Auto-Detect flag turned **on** on the CPC before formatting, and turned **off** again immediately afterwards, using the E)ditor.

Inverted Data is **not** a parameter which the Amstrad normally uses. MFU has the ability to enhance the BIOS routines to handle formats such as Superbrain and Wren which store their data on disc in **inverted format.** This just means that all the data is put through a logical NOT operation - all the zeros are made into ones and vice versa. As far as we are aware, MFU is the **only** program available for the Amstrad which allows you to set up Inverted Data formats and use them as normal from CP/M.

The last two items which are displayed are **Physical Sector Sequence** and **Logical Sector Table.** The former contains a list of sector numbers, **in the order in which they are physically located on disc.** The concept of Physical Sector Skew was mentioned in Section 4; basically it is just a technique to speed up the response time of the disc drive when a sector is read or written, and is completely transparent to both CP/M and programs. Any utility which **formats** a disc, however, **must** know the order in which to write the sectors initially; this information is contained in the Physical Skew Table.

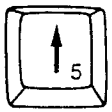
Logical Skew, however, is a more complicated concept. It exists for the same reason as Physical Skew - to speed up accesses to disc - but in this case the sectors are physically recorded on disc in straight numerical order, and a translation of **logical** sector numbers into **physical** sector ID numbers is done by software within the BIOS. Most discs use either Physical **or** Logical skew; there is no advantage in using both. Also, most modern formats use Physical skew because it is simpler. A fuller discussion of Logical Skew was given in Section 5.

Some computers (and those 16-bit machines running MS/PC-DOS in particular) use **no** sector skew of **either** type. This is because they have enough memory to be able to read and write an entire track into or from a memory buffer every time they access disc, and there is thus no need for skew as they do not access individual sectors on the disc.

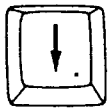
You will notice the marked effect this lack of skew has when the disc is used in a computer such as the Amstrad which **does** access disc a sector at a time; Backup and Verify operations (among others) are noticably slower on such a disc.

6.4 E)dit drive B format

This option displays a reduced set of system parameters pertaining to Drive B - all other entries in the XDPB are calculated automatically by MFU from the values you specify here. The meaning and function of each of the parameters was explained in Section 6.4 above. Selecting the sub-option to EDIT these parameters allows alteration. You may move up and down the list at will, using the



and



cursor keys. **RETURN** also steps down the list without altering the current values. **EXIT** will abort the edit sequence at any time and return you to the Edit sub-menu.

All entries you may make are checked by the editor before being accepted. An illegal entry is rejected and the cursor is returned to the edit prompt on the same line to allow you to enter another value. The checking is done intelligently, with acceptable values depending on previously entered parameters. If, despite this, you still manage to enter a physical format which is invalid then a final check is performed on the whole format at the end of the edit to confirm that there is enough room on the tracks for the number and size of sectors and gaps you have specified.

Some entries must fall within a restricted range to be acceptable. These are as follows :

Sidedness :
0 = Single Sided
1 = Double Sided Std. Cylinder
2 = Double Sided Wrap Round Side
3 = Double Sided Track Order Side
4 = Double Sided Sector Wrap Cyl
All other values are illegal.

Tracks per Side : Must be 80 or less. If a value less **than** 42 is entered, MFU assumes the disc is 48tpi ("40 track"); otherwise, the disc is assumed to be 96tpi ("80 track").

Sector Size : Must be one of 128, 256, 512, 1024 or 2048 (bytes).
All other values are illegal.

Density :
0 = Single Density
1 = Double Density
All other values are illegal.

Block Size : Must be one of 1, 2, 4, 8, 16 (KBytes)
All other values are illegal.

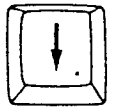
Directory Entries : **Must** be a multiple of 16, up to 256. **Should** normally be a multiple of 32. All other values are illegal.

Gap Width (R/W); :
Gap Width (Format) Consult Appendix B for technical details of the calculations required to determine these values manually. See following paragraph for details of MFU's ability to calculate optimal values automatically.

Auto Format Detect :
0 = On
1 = Off
Any other values are illegal.

Inverted Data :
0 = Normal (non-inverted) data
1 = Invert data on disc transfer
Any other values illegal.

Normally, pressing **RETURN** will step you down the list of of parameters **without** altering them. However, this does not apply to the entry for **Gap Width (R/W)**. If you press **RETURN** when the cursor is on this line, then MFU will assume that you want **both** the R/W and **Format** Gap sizes to be recalculated automatically to optimal values. Thus, if you have altered **either** the number of sectors per track **or** the sector size, you can instruct MFU to perform all the complex Gap size calculations for you. If you select this option, then the newly calculated Gap sizes are displayed. Of course, you are still free to enter a new value for either or both Gaps if you wish to do so. Also, using the



cursor key to step down past the Gap entries will leave them **unchanged.**

Moving the cursor down to the line displaying the **Physical Sector Sequence** will open a second edit screen which will allow you to change either or both of the Physical and Logical skew patterns. For each, you are prompted to alter or leave unchanged; if you select to alter you are presented with a second prompt offering the options of altering the Skew factor, entering a non-standard skew pattern manually, or aborting.

Selecting to enter a new skew factor will prompt for the value; when this is entered the new sector pattern is calculated automatically and displayed. If instead you select to enter a new sequence, you are reminded of the number of sectors per track and the cursor is placed on an edit line. on this line, pressing **RETURN** will step to the right, displaying the current value. Pressing **RETURN** when on or beyond the last entry in the list will exit the edit, with the new list replacing the old. The

cursor keys allow you to move backward and forwards respectively within the list to edit entries.

After you have finished editing the skew, MFU redraws the initial EDIT screen, updated to show the changes you have made. Some internal checks are made at this point, and MFU will refuse to allow you to continue without re-editing if any of the following are found :

1. The overall track format is illegal. This can occur if you have too many sectors and/or they are too large to allow the minimum necessary gaps between them. See Appendix B for a detailed analysis of track format.

2. The computed Group Allocation Vector space is greater than the maximum available. The normal PCW8256/8512 BIOS allows a maximum space for the ALV of 91 bytes, and banked CP/M Plus uses 2 bits per group. This gives you a maximum DSM value of 363 for PCW8256/8512 Auto-Detect type formats. On any type of Amstrad, MFU rearranges the BIOS data areas to allow a maximum ALV space for Drive B of 100 bytes. Thus for all formats on a CPC464/664/6128, and all non-Auto formats on a PCW8256/8512, including those setup by a stand-alone program, the maximum permissible DSM value is 399.

   The maximum permissible DSM can be exceeded if you have a large disc with too small a block size - in particular, 1K block sizes on a disc with over 400K data space (i.e. **not** including reserved tracks but **including** the directory). Increase the block size or reduce the total capacity of the disc.

   If you are not familiar with the concepts and expressions used here, please refer to a good CP/M reference book or article which explains advanced disc allocation theory; this is outwith the scope of this manual. You may also find Section 5 and Appendices B and C of some help.

On completion of edit, you are given the option of returning to edit, aborting or proceeding. If the last is selected, the full XDPB for the new format is displayed and you are given the option of aborting, saving the new format permanently in the library file or installing the new format for Drive B. Note that if you wish to immediately use your modified format then you **MUST** install it using the I option at this point or it will not be retained in memory when you exit.

6.5 F)ormat a disc in Drive B

This option will format a disc **in the format currently assigned to Drive B.** All errors are fully trapped and explained if they occur; an opportunity is given to abort or restart the command.

Before the operation starts, the format in use is redisplayed at the bottom of the screen and you are prompted to insert the disc to be formatted. Press EXIT now if you want to abort.

If you proceed, MFU will format the disc, displaying the number of each track formatted as it does so. Again, press EXIT at any time to abort; however, if you do so **after** the formatting process has started, you must **NOT** attempt to use the partly-formatted disc. It **MUST** be **completely** reformatted before use. If there were previously any files on the disc, they will almost certainly have been destroyed if even just the first 2 tracks were formatted before you stopped it.

When the disc has been formatted, you will be informed and asked whether or not you want to verify it. This will carry out a check of the whole disc and report any physical errors which are detected. If any are found, the disc is damaged and should be thrown away. These discs are cheap - don't risk important programs or data just to save a pound or two. Please see Section **9.11** for a complete description of the **V**)erify operation.

One special case exists, for formats which have the **Auto Format Detect** flag set to **ON** - certain native Amstrad formats, for example.

In this case, before the formatting operation is started you are reminded that the Auto Detect flag is on and given the choice of continuing or aborting. If you continue, then MFU will write the appropriate Format Descriptor Block to disc. This reminder is issued simply as a check for formats which may accidentally have had the Auto Detect flag set to ON.

Further to this, you **cannot** format a disc in a format which has the Auto Detect flag set to ON but does **not** have at least one reserved track to contain the Format Descriptor Block. MFU will detect any attempt made to do so, and abort the format operation with an appropriate error message.

If you are not familiar with the concept of Auto-Format Detection as implemented on the different types of Amstrad computers, please refer to Section 6.3, where the subject is covered in detail.

6.6 L)ibrary manager

This option enters the Library Manager. MFU displays a list of formats already defined within the library, and allows you to perform a range of operations on them.

The specific functions available within the Library Manager are as follows :

6.6.1 I)nstall a format from the library

Select this option if you wish to load a format from the library onto Drive B. Any format shown in the library listing above the menu can be selected by entering its number at the prompt. Pressing **EXIT** at this point will abort and return you to the Library Manager menu. If you enter a number, a prompt such as

Install Amstrad CF2DD format? Press RETURN to confirm, or press EXIT to abort.

>

will be given. The name of the format you have asked to install is shown (Amstrad CF2DD is used here as an example). Pressing **EXIT** at this stage will again abort back to the Library Manager menu without altering the current Drive B format. Pressing **RETURN** at this stage will install the requested format. You will then be returned to the Library Manager menu.

6.6.2 D)elete format from library

This option will delete an existing format from the library if it is no longer required. The procedure to step through the command is identical to that described above for installing a file.

6.6.3 R)ename entry

This option allows you to alter the name of any entry in the library. When you select it, you are prompted for the number of the format entry which you wish to rename. Pressing EXIT at this point will abort back to the Library Manager menu; entering a valid format number will result in the prompt

Renaming Amstrad CF2DD. Enter new name or EXIT to abort.

Newname :

being given. Again, EXIT will abort, or you can enter the new name for the entry. This will result in a message such as

Old entry Amstrad CF2DD now renamed <new name>

being displayed, and after pressing RETURN you will be returned to the Library Manager menu.

6.6.4 V)iew a format

This option allows you to check the contents of an entry without altering either it or the currently installed format in any way. When you select it, you are prompted for the number of the entry to view. Pressing EXIT at this point will abort the command, otherwise the selected format is displayed in exactly the same way as described in Section 6.3

To return to the Library Manager menu, press RETURN.

6.6.5 E)dit an entry

This option allows you to alter a format **without** having to install it first. As with the other Library Options, you are prompted for the number of the format you wish to edit; pressing EXIT at this point will abort back to the Library Manager menu, otherwise the Editor will be invoked and the parameters for your chosen format will be displayed along with the normal Editor prompt to edit, proceed or abort. The operation of the Editor was described in detail in Section 6.4

Once you have completed your alterations, selecting the **proceed** option will display the full list of XDPB values which MFU has calculated from your input, and the prompt

Press 's' to save edited entry, or RETURN to re-edit :

is given. If you wish to overwrite the old library entry with the altered one, select the **save** option; otherwise return to re-edit or abort.

If you abort from the edit process, no change is made to the library entry.

Editing a library entry has absolutely **no** effect on the format currently installed in memory for Drive B.

6.6.6 EXIT to return to Main Menu

Selecting this option will exit the Library Manager and return you to the Main Menu.

6.7 Non-CP/M File Transfer

MFU is capable of reading and writing files from a variety of "alien" (i.e non-CP/M format) discs. It is **not** possible to set up a format such as MS-DOS in memory and use such a disc normally, as we can for a CP/M format. Instead, MFU provides support for copying specific files to and from such discs using this this option.

Version 4 supports two main groups of "alien" formats -

BBC and Master formats (including Acorn DFS and ADFS);

IBM-PC (i.e. MS/PC-DOS formats as used by the PC and compatibles, including the Amstrad PC-1512/1640).

Selecting the appropriate option from the short menu will take you into the corresponding file transfer module, giving more information and displaying sub-options available.

6.7.1 B)BC formats

Selecting this option allows you to transfer files from **CP/M** to **BBC** format discs and vice-versa. Files can be transferred between a 3" PCW disc in drive A (or the PCW's RAM disc) and several types of 5¼" BBC format discs in drive B. The following types of BBC and Master discs are currently supported by MFU :

Acorn DFS	-	**SS/SD 40 track**
Acorn ADFS	-	**DS/DD and SS/DD 40 and 80 track**
Cumana QFS	-	**SS/SD 80 track ONLY**
Opus DDOS	-	**DS/DD 80 track**

You needn't worry if you do not know which of these formats is used by your BBC since MFU automatically attempts to determine if the disc is of a type which is supported, and if so, adjusts itself accordingly.

Please note that both the PCW8256 and the CPC6128 **cannot** normally use **Single Density** formats, and so you are restricted initially to more modern types of BBC discs **only**, such as Acorn ADFS as used on the Master series. However, Appendix F details a small hardware modification to the PCW's Disc Controller circuitry which, if made, **will** allow your computer to use older Single Density BBC formats. Please note that there is unfortunately **no** equivalent simple modification for a **CPC6128**, although a **CPC464**'s FDI-1 interface **can** use both Single and Double Density discs **without** any alteration at all. Vive la incompatabilite - please blame Amstrad for this mess, not us!

When you select the File Transfer option from the main menu, a sub-menu will appear. This offers the choice of transferring files from BBC to CP/M (Option A) or from CP/M to BBC (Option B). Pressing EXIT at virtually **any** point within this option will abort and return you to the Main Menu.

Note that **all** file transfer operations take place using the main, or **root**, BBC directory. This means that files **cannot** be transferred to or from sub-directories with formats such as Acorn ADFS. This restriction is necessary because of the limitations imposed by CP/M, and by older types of BBC formats such as Acorn SD DFS, which do not support sub-directories. In practice this should not be a problem since any DFS which supports sub-directories will also provide the facilities to transfer files between these and the root directory.

Transferring files from a BBC format to CP/M

If you wish to transfer files **from** a 5¼" BBC disc **to** either a 3" disc or into RAMdisc, then select **Option A** from the menu.

You will be asked to insert your BBC source disc into Drive B. When you have done this and pressed RETURN you will be asked to select the side of the disc (A or B) which you wish to copy files from. If the disc is single-sided then enter **A** (or press RETURN). Otherwise enter **A** or **B** as required.

The appropriate side of the disc will then be automatically analysed and its format displayed. If this is one of the BBC formats supported by MFU then the disc directory will be read. Note that some BBC discs contain more than one volume of files - if this is the case with the source disc then you will be asked to select a volume from the list displayed.

Next you will be asked to choose where you want to put the files - this can be either to a 3" PCW disc in drive A or to RAM disc. After selecting the destination drive you will be told how many files are on the source disc and asked if you wish to copy all of these. Answering "N" will cause a list of all filenames to be displayed and you may select which of these you wish to transfer.

After you have chosen the required files then they will be transferred, one at a time, from the BBC disc to your specified destination drive. As each file is copied its name is echoed to the screen to keep you informed of the program's progress.

If any of the files on the BBC disc have a directory code other than '$' then this code will be appended to the filename when it is transferred to CP/M. Thus a BBC file called, say, W.SAMPLE will become SAMPLE.W under CP/M, but a file called $.TEST will become just TEST.

Occasionally, when transferring files, you may find that you run out of space on the destination disc. If this should happen then the transfer will be suspended and the following message will appear:

NO MORE ROOM ON DESTINATION DISC FOR FILE(S)

Insert a new disc in DRIVE A and press RETURN

If you do not wish to continue transferring files then just press EXIT and follow the prompts to return to the main MFU menu.

If you do wish to continue then you must insert a new disc into **DRIVE A - even if the current destination is a RAM disc** - then press RETURN. The transfer process will then be resumed and all subsequent files will be written to the new disc. If you are transferring a lot of data from a BBC disc you may find that you have to use several 3" discs to hold it all. Remember, though, that you can flip each disc over and use the other side when the first side becomes full. If you are transferring a particularly large data file you may find that it just will not fit on a 3" disc at all, in which case you will have to abort the transfer by pressing EXIT.

If you have a PCW with a full-capacity, 368K RAM disc then this should be sufficient to hold even a large BBC file since a standard Acorn DFS disc has a maximum filesize of 64K. Some more modern DFSs, however, do allow much larger files. Once you have such a file in the RAM disc you can then transfer it - using PIP - to a large-capacity CP/M format 5¼" disc in drive B.

Transferring files from CP/M to a BBC format disc

This option allows you to transfer files **from** a 3" **CP/M** disc in drive A to a 5¼" **BBC** disc in drive B. (Alternatively, you may use drive M as the source.) The sequence of operations which must be carried out is very similar to that used to transfer files from BBC to CP/M, except that the source and destination drives are reversed.

The process is initiated by selecting option B from the File Transfer Menu. You will then be prompted to select the source drive (A or M) containing the files which you wish to transfer. Next you will be asked to insert the destination disc into drive B. This disc should be formatted to one of the standard BBC formats **BUT SHOULD OTHERWISE BE BLANK.** (This is not strictly necessary but any existing files on the disc will be over-written by the transferred files thus it is safer to use a completely blank disc.)

When you have inserted the disc you will be prompted to enter a filespec and also the CP/M user number where your source files are to be found. This will usually be User 0, in which case you may simply press RETURN, but otherwise you must enter the user number (1 to 15) then press RETURN. After this has been done you may select individual files for copying, as in the previous section, or copy all those files which match your filespec. You are then given the opportunity to add a volume label to the BBC disc if desired. The program will then proceed to transfer the chosen files to the BBC disc.

If you are transferring files to a small-capacity (single-sided and/or single-density) BBC disc then there may not be enough room on this to hold all your files. Should this situation occur then the program will transfer as many files as possible and then inform you that there is no more room on the destination disc. If you wish to transfer the remaining files then you will have to repeat the operation using a new disc and select for copying those files which have not yet been transferred.

With larger capacity BBC discs which are capable of holding more than one volume of files then volumes will be allocated as required. Thus if you are transferring a lot of files then you may find that not all of them will be located in Volume A - the remainder will be placed in Volumes B,C,D and so on.

6.7.2 I)BM-PC (MS/PC-DOS) formats

Selecting this option allows you to transfer files from **CP/M** to **MS/PC-DOS** and vice-versa. Files can be transferred between a 3" PCW disc in drive A (or the PCW's RAM disc) and **ANY** standard 5¼" MS/PC-DOS disc in drive B. The following types of MS/PC-DOS discs conform to the Microsoft standard and are currently supported by MFU:

Single-sided 40-track with 8 sectors per track
Single-sided 40-track with 9 sectors per track
Double-sided 40-track with 8 sectors per track
Double-sided 40-track with 9 sectors per track

You needn't worry if you do not know which of these discs is used by your PC since MFU automatically adjusts itself according to the type of disc used.

When you select the File Transfer option from the main menu, a sub-menu will appear. This offers the choice of transferring files from MS/PC-DOS to CP/M (Option A) or from CP/M to MS/PC-DOS (Option B). Pressing EXIT at virtually **any** point within this option will abort and return you to the Main Menu.

Note that **all** file transfer operations take place using the **root** MS/PC-DOS directory. This means that files cannot be transferred to or from sub-directories. This restriction is necessary because of the limitations imposed by CP/M, and by some earlier versions of MS/PC-DOS, which do not support sub-directories. In practice this should not be a problem since any PC which supports sub-directories will also provide the facilities to transfer files between these and the root directory.

Transferring files from MS/PC-DOS to CP/M

If you wish to transfer files **from** a 5¼" MS/PC-DOS disc to either a 3" disc or into RAMdisc, then select **Option A** from the menu.

You will be asked to insert your MS/PC-DOS source disc into drive B. When you have done this and pressed RETURN the disc will automatically be analysed and its format displayed. Next you will be asked to choose where you want to put the files - this can be either to a 3" PCW disc in drive A or to the currently assigned RAM disc. After selecting the destination drive you will be asked to enter a **filespec** for the files you wish to copy. This may be anything from a single filename, such as DATABASE.DAT, to a filename containing **wildcards**, such as ***.DOC** (this would specify **ALL** files on the disc which have the extension **.DOC** - see your PCW User Guide for a full explanation of wildcards in filenames). Simply pressing RETURN will allow you to specify all files on disc.

The directory of the MS/PC-DOS disc will then be scanned to find if there are any files on disc which match your specification. If so then the number of matching files found will be displayed and you will be asked if you wish to copy them all. Answering "N" will cause a list of all matching filenames to be displayed and you may select which of these you wish to transfer.

After you have chosen the required files then they will be transferred, one at a time, from the MS/PC-DOS disc to your specified destination drive. As each file is copied its name is echoed to the screen to keep you informed of the program's progress.

Occasionally, when transferring files, you may find that you run out of space on the destination disc. If this should happen then the transfer will be suspended and the following message will appear:

NO MORE ROOM ON DESTINATION DISC FOR FILE(S)

Insert a new disc in DRIVE A and press RETURN

If you do not wish to continue transferring files then just press EXIT and follow the prompts to return to the main MFU menu.

If you do wish to continue then you must insert a new disc into **DRIVE A - even if the current destination is a RAMdisc** - press RETURN. The transfer process will then be resumed and all subsequent files will be written to the new disc. If you are transferring a lot of data from a MS/PC-DOS disc you may find that you have to use several 3" discs to hold it all. Remember, though, that you can flip each disc over and use the other side when the first side becomes full. If you are transferring a particularly large data file you may find that it just will not fit on a 3" disc at all, in which case you will have to abort the transfer by pressing EXIT.

If you have a PCW with a full-capacity, 368K RAM disc then this should be sufficient to hold even the largest MS/PC-DOS file since a standard 40-track MS/PC-DOS disc can only hold a maximum of 360K. Once you have such a file in the RAM disc you can then transfer it - using PIP - to a large-capacity CP/M format 5¼" disc in drive B.

Transferring files from CP/M to MS/PC-DOS

This option allows you to transfer files **from** a 3" **CP/M** disc in drive A **to** a 5¼" **MS/PC-DOS** disc in drive B. (Alternatively, you may use the currently assigned RAMdisc as the source.)

The sequence of operations which must be carried out is very similar to that used to transfer files from MS/PC-DOS to CP/M, except that the source and destination drives are reversed. The process is initiated by selecting option B from the File Transfer Menu. You will then be prompted to select the source drive (A or RAMdisc) containing the files which you wish to transfer. Next you will be asked to insert the destination disc into drive B. This disc should be formatted to one of the standard MS/PC-DOS formats **BUT SHOULD OTHERWISE BE BLANK.** (This is not strictly necessary but any existing files on the disc will be over-written by the transferred files thus it is safer to use a completely blank disc.)

When you have inserted the disc you will be prompted to enter a filespec, as in the previous section, and also the CP/M user number where your source files are to be found. This will usually be User 0, in which case you may simply press RETURN, but otherwise you must enter the user number (1 to 15) then press RETURN. After this has been done you may select individual files for copying, as in the previous section, or copy all those files which match your filespec. You are then given the opportunity to add a volume label to the MS/PC-DOS disc if desired. The program will then proceed to transfer the chosen files to the MS/PC-DOS disc.

If you are transferring files to a small-capacity (single-sided) MS/PC-DOS disc then there may not be enough room on this to hold all your files. Should this situation occur then the program will transfer as many files as possible and then inform you that there is no more room on the destination disc. If you wish to transfer the remaining files then you will have to repeat the operation using a new disc and select for copying those files which have not yet been transferred.

6.8 P)roduce stand-alone Format setup Program

This option allows you to create a small machine-code program on disc which, when run, will set up a specified format on Drive B. Note that the format **currently assigned** to drive B is written out to this program when it is created. This means that if it is wished to create a program to set up a different format to that currently in use then the other format must first be loaded using the **L)**ibrary Manager, or created using the **E)**ditor.

On entry to this option, you are reminded of the above and given the choice of proceeding or returning to the Main Menu. If you proceed, you are requested to enter the filename of the program you wish to create. We suggest that this bears some resemblance to the name of the format in the library to avoid confusion - for example, you might use **AMSCF2DD** as the name for a program to restore the default **AMS**trad **CF2DD** format.

Press EXIT at **any** point during the entry of the filename to abort and return to the Main Menu.

If you have a requirement for one or more formats to be installed onto Drive B frequently (perhaps one which you would like as a default when you boot) then you should use this option to create the necessary program files. These take up much less disc space than the MFU system, are wholly independent of it and save the time and hassle of running MFU itself every time you want to install a format.

To install a format as the default on boot, create a stand-alone program using this option, copy it onto your boot disc and then include its name in your PROFILE.SUB autoexec file.

Stand-alone format setup programs produced by MFU must **NEVER** be run on a different type of computer to that on which they were produced, **OR** run under an older BIOS version than that under which they were produced. Any attempt to do either of these things will crash the computer.

6.9 S)et system parameters

This option allows you to alter some of the parameters MFU uses to perform various functions.

The Main Menu within the Setup module lists the main subjects which have configurable options. These are a subset of the configuration parameters which can be altered with MFUPATCH.COM (see Appendix A). If you need to alter any of the options frequently, then you would be better configuring a second version of MFU using MFUPATCH.COM; the **S)**etup option is only really useful for quick fudges to things like the printer control strings or drive speed tolerance for a special reason.

The four main subjects are:

1. Printer
2. Drives
3. Files and Overlays
4. Miscellaneous

6.9.1 Altering Printer defaults

There are five parameters concerned with the printer which can be altered. These control the command strings which are sent to the printer when MFU is told to dump the screen. The defaults which are initially installed into MFU perform the functions listed below for the PCW's Epson(ish!)-compatible printer.

When you ask to alter a printer string, MFU will display the current contents in **hex** and will prompt you for replacement input. If you simply press RETURN at this point, the string will be emptied. Otherwise, you may type in up to 15 bytes in hex, separating each with a space, tab or comma. Within a byte, DELete allows you to correct a mistake. You cannot move backwards in the string or forwards other than by entering values.

The command strings you can alter are as follows :

1. Mode

The default is normally for screen dumps to be done in DRAFT mode for speed. If you require a higher-quality dump for a permanent record of a format, use this option to change the mode from DRAFT to NLQ.

2. Command string to initialise printer

This string of bytes can be sent to initialise the printer to any desired configuration of character pitch and style, page length etc. You can alter the contents to allow any type of printer to be used. Up to 15 bytes can be sent as an initialisation string; the default string installed into MFU sets the page length to 70 lines (A4).

3. Command string for printer reset

This string is used to restore the printer to a default state after a screen dump has been made. So, if you want, you can use a peculiar printer mode to do your dumps without affecting the normal setup of your printer. The default string is EMPTY.

4. Command string to set NLQ mode

This is the string sent after the initialisation string when the printer is to be set to NLQ mode.

5. Command string to set DRAFT mode

This is the string sent after the initialisation string when the printer mode is to be set to DRAFT.

6.9.2 Configuring Drive parameters

This option allows you to alter the two main parameters which are affected by the type of 5£" drive you have attached to your Amstrad, as well as specify the drive designation of your RAMdisc.

1. Drive Type

MFU will always adjust itself and the Amstrad automatically to take account of the type of 5£" drive you have, but you must tell it! The default is for a **96tpi** (80-track) drive, which also covers 40/80 switchables; if you want to use a **48tpi** (40-track only) drive, then alter this parameter accordingly.

For normal operation of MFU - EVEN WHEN USING A 40 TRACK FORMAT - Drive Type should be left set to 80 TRACK.

MFU will **automatically** compensate and force the BIOS to double-step when a 40 track format is installed with an 80 track disc drive. There is **NO** need normally to alter this parameter - though doing so will do no harm. For convenience, if you have a switchable drive, leave both this parameter and the disc drive set to **80 track.**

2. Drive Speed tolerance

This is a percentage value, representing the maximum variation in the rotational speed of a disc in Drive B. If you do not know what the value is for your particular drive, we suggest using 2%, which is the default value. This figure is used throughout MFU to determine the sizes of Gaps 3 and 4 necessary to compensate for drive speed fluctuations while formatting a track or writing data to the disc. If you are not familiar with this concept, please consult Section 4 and Appendix B before altering this parameter.

3. RAMdisc designation

This option allows you to specify the CP/M designation of the temporary drive MFU will use as a buffer during file and disc copying operations. On a PCW, this will normally be configured to be **Drive M**; a CPC version of MFU will normally have this default set to **None.**

On a CPC, you might want to change this if you have a third-party add-on RAMdisc (such as DK'tronics) and want the same benefits of speed and temporary storage capacity that PCW owners have. Normally on a CPC, MFU will use a blank disc in Drive A as a buffer. For MFU to be able to use a RAMdisc on a CPC, the appropriate CP/M drivers **must** already be installed; MFU will access it **only** by official high-level BDOS function calls.

On a PCW, you would normally never want to alter this parameter. However, if you have a hard disc attached to your machine (as Drive C, perhaps) then you could choose to use this as MFU's buffer - it will be almost as fast as the normal RAMdisc, but is unlikely ever to fill up during a copying operation! This means that making backups of discs, for example, will not require any disc-swapping.

Any valid drive letter can be chosen for your RAMdisc buffer from the range **A** to **M.** This should obviously be a valid drive - i.e. you must have a drive of that name present on your computer! If you change this parameter to an illegal drive, all of MFU's copying functions will detect this and will abort.

6.9.3 Configuring Files and Overlays

This option allows you to change the file that MFU will use to hold its format library, and the drive on which MFU will look to find its overlay files.

1. Name of Library Data file

Each library file has a maximum capacity of 60 formats on a PCW and 39 formats on a CPC, so this facility is useful if you fill the main library. You can make a copy of the main library and delete most or all the formats from it to give you a new, empty one to work with; use this option to change the name of the default library to your new file when you want to use a format from it or add a new one to it.

2. Drive used to hold overlays

The drive where MFU expects to find its overlay files is normally the logged drive (represented here by the **@** symbol). If for any reason you want to temporarily change this to a named drive, you can do so using this option. Note, however, that if you do so then the overlays **must** be present where you say they are, or MFU will object!

6.9.4 Configure miscellaneous options

Currently, there is only one other option you can alter; the **Default Skew Factor** which is assumed by MFU when it cannot identify a format in memory from its library. This skew factor is then used for physical skew when displaying or editing the format's parameters or formatting a disc.

A value of 2 is an average which will be acceptable for most computers; if however you want to alter this (say, for example, you are playing about with a skew-less DOS format) then you should use this option to change the default skew factor accordingly.

The default is never used for **logical** skew; MFU can detect if logical skew is active from the BIOS disc parameter header data blocks, and displays and uses this information as necessary.

6.10 T)rack-by-Track backup of a disc in Drive B

This option will perform a Track-by-Track physical backup of a disc in Drive B. The whole disc will be copied onto another; any information on the second (target) disc will be overwritten and destroyed.

Both the disc to be copied **and** the second, target disc **MUST** be of the format currently assigned to Drive B.

During the copying process, MFU will use the currently assigned RAMdisc as a buffer if you have one. Otherwise, you will be prompted to insert a blank 3" disc into Drive A for use as a buffer. No files already on the buffer drive will be altered or damaged by this; however, it is obviously a good idea to have as few files as possible there to give MFU the largest possible buffer space, and so reduce the number of times discs will have to be swapped in Drive B.

Before starting, MFU works out how much space is free on the buffer drive and displays it, along with the number of tracks from the disc to be copied which this size of buffer can hold, and the total number of tracks to be copied. This lets you see how many disc-changes will be required to copy the disc; if it seems too many then press EXIT to abort, exit from MFU and move or delete files to free more space for the buffer.

If there is less than **64K** free on the buffer drive, MFU will abort the backup at this point as a smaller buffer would involve a ridiculously large number of disc exchanges in Drive B when copying any relatively high-capacity disc. Move or delete files to make room for a more realistic buffer, and try again.

When you proceed with the copy, you will be prompted to insert SOURCE and DESTINATION discs alternately as the operation proceeds. **ALWAYS WRITE-PROTECT THE SOURCE DISC BEFORE YOU BEGIN THE COPY** - this prevents the source disc being damaged if you accidentally mix up the two discs during the operation.

You can abort at any time by pressing EXIT. If you do this before the disc has been fully copied, you must **NOT** attempt to use the partially-copied destination disc. Either erase all files which appear to be on it (most will either **not** actually be present or will be corrupt), restart the Backup again or reformat the disc.

When the copying process is complete, you will be informed and asked whether or not you want to verify the destination disc. This will **not** compare the two discs, but will simply carry out a physical check of the destination disc to ensure that there are no physical defects. Please refer to section **6.11** for a full description of the **V**)erify operation.

6.11 V)erify disc in Drive B

This option will physically verify a disc in Drive B, comparing its structure with the format currently assigned to Drive B. All errors are fully trapped and reported, with options to abort or retry. Please note that the actual **data** on the disc is not being checked as such, only its integrity.

The Verify operation does **not** damage or alter information stored on the disc in any way.

As the Verify operation proceeds, the numbers of the track and side currently being checked are displayed. The speed at which a disc will be verified depends on the number of sectors per track and their skew. Discs with a skew which is inappropriate for the PCW (such as MS/PC-DOS discs) will verify relatively slowly. This does **not** mean that there is anything wrong with them.

You can abort at any time by pressing EXIT.

If errors are detected by the Verify operation, then there is no alternative to reformatting the disc. If you have any important files on the disc which you don't have copies of elsewhere, try to copy them onto another disc first using a file copying program such as **PIP**. If, after reformatting, the disc will still not verify without errors, throw it away! Discs are cheap - programs and important data are not. Don't risk losing something important just to save a pound or two.

6.12 EXIT to CP/M

Pressing EXIT when in the Main Menu will produce the question

Exit to CP/M (y/n)? :

to confirm that you wish to leave MFU.

Responding to this with "Y" will return you to the operating system. The format currently assigned to Drive B will not be altered on exit.

Be sure that Drive B is set to the format you want before exiting MFU.

APPENDIX A

Customising MFU with MFUPATCH.COM

MFU has been designed to allow external alteration of any of the default S) options, as well as several other parameters such as terminal control strings.

The program MFUPATCH.COM is supplied with the MFU system to allow permanent customisation of a copy of MFU to your own requirements.

MFUPATCH.COM appears in use to be very similar to the S) option within MFU. This is not accidental - most of the code is the same! MFUPATCH.COM does however allow you access to several parameters not accessible from within MFU. Also, of course, MFUPATCH.COM alters the actual defaults within a copy of MFU, so is useful if you find yourself having to use the S) option a lot.

You create customised copies of the system by configuring **only** the MFU.COM root program. As this is less than 8K in size, you can have several different copies of it on a disc, all of which are configured for a different job but still use the **same** large overlay files.

When you enter MFUPATCH.COM, you are asked for the name of the file you wish to customise. This should be a copy of MFU.COM. Once in MFUPATCH.COM, you should always check that the copy of MFU you are working on "knows" its new name using the option detailed below to do so.

Most of the options available in MFUPATCH.COM are the same as those described in Section 6.9, and these will not be covered here. Important extra options are as follows :

A.1 Configure Terminal

MFU can send control strings to the VT52 terminal emulator; if you want to have a special setup (perhaps you prefer to work in inverse video, for example) then you can configure MFU to do this for you.

1. **Terminal Initialisation string**

   In a similar way to the printer, MFU send an initialisation string to the terminal emulator when it starts to run. The default string is a "Home and Clear" command - 27h, "H", 27h, "J".

2. **Terminal Exit string**

   MFU can also send a command string to the terminal emulator when you quit the program. If you want to ensure that the terminal is left in a particular state, then configure this string accordingly. The default string is a "Home and Clear" command - 27h, 48h, 27h, 4Ah.

A.2 Configure MFU program filename

MFUPATCH.COM gives you access to an extra parameter in the **Files and Overlays** option - you can change the name of the main program file (normally MFU.COM) if you are configuring a copy with a different name. If you are configuring a file whose name is **not** the same as that displayed when you enter this option, you **must** change the imbedded default name to be the same as the actual file name of the root program copy you have created or it will not be able to run correctly.

APPENDIX B

Track Format and Gap Equations

This appendix is technical and descibes the physical layout of a track on a disc.

B.1 Track Layout

Almost all modern discs (including Amstrad's) use IBM track layout. This should **not** be confused with the various logical and physical disc formats used by IBM computers - the term is used here to describe the now standard way in which data is physically recorded on a track by the Disc Controller hardware.

Under the IBM system, each track is laid out with a Track Header marking the start of the track, and a number of sectors separated by **gaps**. Each sector also has a header, describing the number and size of the sector and providing checksums of the address and data information stored in it.

All of these headers and gaps **must** contain particular information and occur in the correct order if the Disc Controller is to be able to read the track.

There are one or two other (and less common) track layout systems around. Of these, the only one which looks likely to ever amount to anything is **ISO** (sometimes instead called after **Sony**, who invented it) **ECMA** format. This has been developed recently to allow modern high-capacity discs to be used to their full capacity - the older IBM system, though a standard, is rather wasteful. Much of the information stored in the track header, for example, is not really needed by modern Disc Controller chips and the space it takes up could be better used to hold more data! However, the IBM system has become such a standard that ECMA will have an uphill struggle to be accepted. The NEC uPD765A FDC chip used in the Amstrad **is** partially capable of reading ECMA format discs; however, there are serious restrictions on this and if you require this capability you should consult either ourselves or NEC for assistance.

To complicate matters further, there are **two** different fundamental techniques used to actually write information onto the magnetic media. The first of these is **Frequency Modulation** (**FM**), and is used on **Single Density** discs. This dates from the time discs were first used on computers, and is no longer common unless you are unfortunate enough to own a BBC.

The second, and newer, system of information recording is known as **Modified Frequency Modulation**, or **MFM**. This is used by **Double Density** discs. As the name suggests, this technique packs **twice** as much information onto each track as the older Single Density method. MFM is nowadays used almost exclusively.

Due to the design of the Disc Controller circuitry on PCW8256/8512s, they **cannot** normally use Single Density discs. However, MFU **is** capable of handling and understanding Single Density formats, and Appendix F describes a simple set of hardware modifications to the PCW's main circuit board which **will** allow it to use **both** types of disc.

CPC464 owners have an advantage over everyone else here - they **can** use Single Density formats **without** any hardware modifications! Unfortunately, however, there are **no** equivalent modifications possible for the CPC6128. We don't know if it is possible for the CPC664 to read single density discs as we can't find one to try it on! If you have one of these rare beasts, please get in touch!

As has been mentioned previously, the sectors on a track are separated by **gaps**. These are needed to compensate for variations in the rotational speed of the disc as data is written. If there were no gaps, any slight fluctuation in speed while the data was being written could result in the start of the next sector being overwritten accidentally. These **inter-sector** gaps are referred to as **Gap 3**. Another gap (**Gap 4**) is used to compensate for an overall speed variation when the whole track is written during the formatting process. If this gap was not present, there would be a risk of the last sector written accidentally destroying the first, or at least encroaching into the track header.

As track layout is slightly different for the two types of disc (FM and MFM), it is necessary to develop two different sets of equations for calculating the optimum gap sizes for a given disc format. MFU can do this automatically when a format is defined or altered using the E)ditor function; the following section derives the appropriate equations for manual calculations.

B.2 Track Equations for 5¼" floppy disks

B.2.1 Total track capacities :

These are calculated from the data rate per second and the rotational speed of the disc. For a standard 5¼" disc, the rotation rate is 300 revolutions per second, and data is read from and written to the disc at the rate of 125 thousand bits per second (Single Density FM) and 250 thousand bits per second (Double Density MFM). Thus the total unformatted track capacities for the two systems are as follows :

Single Density : 125Kbits/sec @ 300rpm = 3,125 bytes

Double Density : 250Kbits/s @ 300rpm = 6,250 bytes

B.2.2 Track layout :

Firstly, we must determine the minimum size acceptable for Gap 4 to compensate for drive speed variation while formatting a track. **Gap 4a**, at the start of the track, is a fixed size -

Gap4a = 40 bytes (FM) 80 bytes (MFM)

Gap 4b is used to actually compensate for drive tolerance on the track, and so it is

Gap4b (min) = Drive tolerance (%) * 3,125 for FM, and

Gap4b (min) = Drive tolerance (%) * 6,250 for MFM.

Gap4 = Gap4a + Gap4b

Using this figure, along with values for the total sizes of track and sector headers, as well as the number of sectors per track and their size, we can calculate both the minimum necessary inter-sector gap size for Gap 3 and the maximum space which is available for them.

Track header = 33 bytes (FM) 66 bytes (MFM)

Sector Header = 33 bytes (FM) 62 bytes (MFM)

Number of Sectors per Track = NumSec

Sector Size in bytes = SecSize

Thus general equations for both densities are

Gap3 (min) = (SecSize + SecHdr) * Drive Tolerance (%)

Gap3 (max) =

((TrkCapacity - Gap4 - TrkHdr) / NumSec) - SecHdr - SecSize

If Gap3(min) is greater than Gap3(max), then we have an illegal format - there is not enough room on the track to hold even the minimum necessary Gap3 to allow the sectors to be reliably formatted.

The minimum size for Gap3 which we have calculated is that which will allow a sector to "move" forward or backwards safely on a track when it is written.

This value is known as **Gap 3 (Format)**, and represents the **total** space left between two adjacent sectors.

When the Disc Controller writes a data sector onto the disc, it writes blank Gap information at the end of it to cover up any confusing remnants of the previous contents of that sector. However, as the sector we are writing may be "late", and the next sector on the track may be "early" due to speed variations in the drive, the controller can only safely write around **half** the length of Gap 3 (Format) after any sector. The size of this **Read / Write** gap must be specified whenever we issue a low-level Read or Write command to the FDC, and as we have seen is calculated simply as

Gap3 (R/W) = Gap3 (Format) / 2

B.3 Technical references

Physical Track formats and recording methods are extremely complex subjects, and more detailed discussion of these is outwith the scope of this manual. If you require further information on either these subjects, or low-level internal details of the uPD765A FDC chip, you should consult the following publications, available from NEC Electronics (Europe) GmbH or main U.K. distributors.

uPD765A/uPD7265 FDC data sheet 765A/7265DS-Rev1-7-83-CAT-L
(or more recent revision if available)

uPD765A/A-2 Floppy Disc Controller Product Description

Application Note **HCP1** by Kohl and Westerdorf

(Application Note **HCP2** by same authors covers **Sony ECMA** format; if you require this make sure you also get a **recent** uPD7265 data sheet and/or Product Description.)

Appendix C

CP/M Directory structure

This appendix is technical, and describes the layout, content and use of the CP/M directory structure on a disc.

In order to impose a useful data structure on a disc, CP/M puts our data into files. These files can be located **anywhere** on the disc - the data within a file need not even be laid out contiguously on the disc. Data can be put into **any** free data group; however, CP/M keeps track of where each one is, and the order in which they were used, in the **directory** for the disc.

The directory occupies the first few groups at the start of the data space; obviously, the space it occupies is reserved and can never be used for holding our data. However, directories are normally quite small compared to the overall capacity of a disc - usually around 1%, in fact.

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.

```
0053494420202020  20434F4D0000003A    .SID     COM...:
333435363738393A  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 is stored.

The first byte contains the user number of the file; a value of 0E5h means that the file has been deleted. (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 0E5h. CP/M will then re-use the data groups belonging to the deleted file as and when it needs them. It is only once the groups have been re-used for something else that their data is lost, so UN-ERAse programs **can** recover accidentally deleted files - provided nothing new has been written to the disc in the meantime!

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

Bytes 16 to 32 are the "index" which tells us where to find the **data** for the file - each byte is the number of a **block** which has been allocated to the file. As we saw previously, CP/M on the Amstrad 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 the 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.

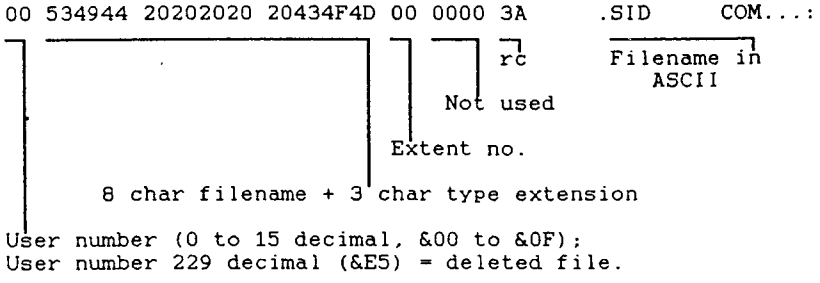
The **Record Count** (byte 15) is the exact number of records which contain valid data held within the blocks pointed to by an extent. As an extent can index up to 16k, and a record is 128 bytes in size, then obviously the maximum number of records held by an extent is 16 x 8, or 128 (80h). Thus extents of large files which are completely filled will have record counts of 80h; 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.

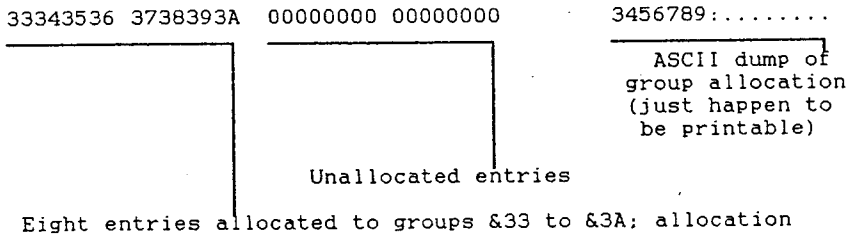
Also, because the first two extents have been completely filled, in this example, **all** sixteen of their allocation numbers will be used and their **record counts** (byte 15 of each) will be set to 128 (&80). The third extent will contain the allocation information for the remaining 14K of the file, 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 groups in the extent (the last group 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 may make all this a bit clearer :

First 16 bytes (File identification and size information)



Second 16 bytes (Data Group Allocation numbers)



Appendix D

CP/M Disc Parameter Block and XDPB

This appendix is technical, and describes CP/M's internal disc parameter data structures.

D.1 CP/M Disc Parameter Block

The DPB is a block of data, 17 bytes long, which is maintained by the BIOS for use by both CP/M itself and other programs when trying to work out where particular data is situated on the disc. The layout of the DPB is as follows :

SPT	2 bytes	Logical Sectors Per Track
BSH	1 byte	Block SHift
BLM	1 byte	BLock Mask
EXM	1 byte	EXtent Mask
DSM	2 bytes	Data Space Maximum
DRM	2 bytes	Maximum no. of directory entries
AL0	1 byte	ALlocation vector byte 0
AL1	1 byte	ALlocation vector byte 1
CKS	2 bytes	CheckSum Vector Size
OFF	2 bytes	Data area OFFset
PSH	1 byte	Physical sector SHift
PHM	1 byte	PHysical sector Mask

There is a **separate** DPB present for each disc drive (including the RAM-disc). This allows us to use discs with radically different formats in different drives at the same time. MFU gives you direct access to the DPB for the Amstrad's Drive B.

We'll explain each of the DPB entries in order.

1. SPT

The value held here specifies the number of **logical,** 128 byte "sectors" (i.e. RECORDS) which each track on the disc can hold. For Amstrad's CF2DD format, this is 36.

This is calculated simply by multiplying the number of pysical sectors per track by the number of logical records which a sector can hold. So, we have 9 sectors per track, each of 512 bytes and therefore capable of holding 4 x 128 byte records, giving

9 x 4 = 36 Records Per Track

2. BSH and BLM

These values tell CP/M the size of each of the data groups, or BLOCKS, on the disc. As we mentioned earlier, a block can be 1K, 2K, 4K, 8K or 16K in size. The values of BSH and BLM which correspond to each block size are shown in the table below:

Block Size (Kbytes)	BSH	BLM
1	3	7
2	4	15
4	5	31
8	6	63
16	7	127

3. DSM

This value is one less than the total data storage capacity of the disc, measured in blocks. This total **excudes** space on reserved tracks, but **includes** space used to hold the directory.

4. DRM

This value is one less than the total number of directory entries which the directory can hold. Each directory entry takes up 32 bytes, and so the value of DRM must be calculated once the amount of space allocated to the directory is known.

The directory of a CP/M disc can have any number of entries up to 256; however, the number **must** be a multiple of 16, and in standard CP/M systems is normally a multiple of 32.

5. AL0, AL1

These two bytes must be considered together, and specify the number of data blocks on the disc which are reserved for use by the directory. Just to confuse everyone, CP/M puts these bytes together to form a 16-bit word in the **opposite** order to normal 8080 and Z80 practice (i.e. Hi - Lo instead of Lo - Hi).

So we consider the word as shown below :

	AL0							AL1							
Bit :	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

One bit of this word is set to a one for **each** data block which is reserved for the directory, starting from the left of the word (i.e. the most significant bits).

So, if we had 4 blocks reserved for the directory (as in Amstrad CF2DD format), this word would have the value

1111 0000	0000 0000
AL0	**AL1**

i.e. AL0 = &F0 (240 decimal) and AL1 = &00.

Once we know the number of directory blocks from AL0/1, and the size of each block from BSH/BLM, we can calculate DRM as mentioned above (the number of 32-byte directory entries which the directory can hold).

6. CKS

This value specifies the amount of space required (in bytes) for the Directory Checksum Vector for the disc.

In order to make sure that we don't change discs when we're not supposed to (in the middle of writing a file, for example), CP/M "logs in" a disc each time we start reading or writing a file and then checks that the disc is the same each time we read or write data to that file.

The disc is identified by a series of **checksums** which CP/M automatically constructs from the first directory entry in each directory record. These checksums are each 1 byte in size, and there are 4 x 32-byte directory entries per 128-byte record, and so the **Checksum Vector** space required to store them in memory is given by

CKS = (DRM + 1) / 4

Note that the directory checksumming function is performed entirely automatically by the BDOS, transparently to programs.

7. OFF

This value simply specifies the number of tracks on the disc which are reserved for use by CP/M and cannot be used to hold data. As mentioned earlier, these **Reserved Tracks** are normally used to hold the code which makes up part of CP/M itself.

Reserved tracks are always contiguous from and including Track 0. CP/M then places the directory in the first one or more data groups on the disc, starting at the beginning of the first non-reserved track.

8. PSH and PSM

These two values tell CP/M what size our **physical** sectors are on the disc. The values corresponding to the different possible physical sector sizes are shown in the following table:

Sector Size (bytes)	PSH	PSM
128	0	0
256	1	1
512	2	3
1,024	3	7
2,048	4	15

9. EXM

We have deliberately left discussion of EXM till last! If your brain already hurts, you might prefer just to turn the page quickly and forget that you saw this...!

All right, you were warned!

In Section 8.1, we saw that each directory entry has 16 bytes which it uses to hold the Block numbers of the blocks which are allocated to the file to which the directory entry refers. Thus (in theory!) each directory entry can hold allocation information on 16 blocks. The original CP/M block size was 1K, which ties in nicely with the maximum value of the Record Count (rc) byte of the directory entry - &80 (128 decimal) records is 16K of space; i.e. 16 x 1K blocks!

Unfortunately, time moved on and discs got bigger! To allocate space on a disc with a format such as Amstrad CF2DD by allowing only 16K per extent would require a huge number of directory entries - half the disc space would be taken up by its directory!

To get round this, more recent versions of CP/M have introduced Block sizes greater than 1K (specified in fact by the BSH and BLM values in the DPB which we looked at earlier). However, it is still necessary to "fool" CP/M programs into thinking that we are still using the old, 16K extent system!

To do this, CP/M intoduced the tortuous sytem of **extent folding**, where the old type of (**logical**) 16K extents used by programs are "folded into" the actual, **physical** extents which are represented in the disc's directory entries.

This is the fuction of EXM! This value allows CP/M to calculate how many **logical** extents are contained in each **physical** one, and is derived by considering both the size of the blocks used on the disc and the total number of these that the disc can hold (i.e. DSM). This relationship is summarised simply in the following table; the entry in

the table corresponding to the appropriate values of Block Size and DSM is the required value for EXM.

Block Size (K)	DSM < 256	DSM > 255
1	0	Invalid
2	1	0
4	3	1
8	7	3
16	15	7

If you remember, the original specification for a directory entry which we gave in Appendix C said that each of the 16 allocation bytes each contained the group number of one of the data blocks which was allocated to the file. However, this simplistic approach runs into problems when we consider a disc containing **more** than 256 data blocks, as a single-byte allocation number can only have the range 0..255.

So, for example, an Amstrad CF2DD disc with its large (by CP/M standards!) capacity of 720K has 357 data blocks, each containing 2K! The solution is fairly obvious - with such a disc, **two byte** allocation numbers are used. This means that the 16 bytes of allocation information in a directory extent can only hold pointers to **eight** data groups, but CP/M takes account of this automatically because of the distinction made in the derivation of EXM between discs with 256 blocks or less and those with more.

You will be able to see quite clearly how double-byte allocation numbers are stored by looking at the directory of a CF2DD (or other high-capacity) disc using MFU's A)nalysis option. Compare this to the single-byte system used on a lower capacity disc, such as a Wren SS/DD, which was discussed in the previous section.

The following diagram shows the dump of a directory entry for a higher capacity disc with 2-byte allocation numbers which corresponds to that shown previously in Appendix C for the single-byte allocation system :

```
0053494420202020  20434F4D0000003A    .SID     COM...:
3300340035003600  3700380039003A00    3.4.5.6.7.8.9.:.
```

Note that two-byte allocation numbers are represented in conventional 8080/Z80 order (Lo-Hi), where the **lower order** byte occurs **first**, followed by the higher-order byte, rather than in the order they would normally be written. So, for example, group number &0034 is allocated by placing its number into the extent in the order &34,&00.

D.2 Amstrad XDPB

The full specification for CP/M Plus includes some extremely powerful facilities for adding **device drivers** to the operating system. These are routines which control, at the lowest possible level, special "add-on" hardware - for example, a multi-format disc system!

Unfortunately, for reasons know only to themselves, Amstrad do **not** supply a version of CP/M Plus which meets this specification. Instead, programs like MFU have to take over many of the normal BIOS disc functions and replace them with their own, customised routines - the main reason for MFU's size and code complexity.

To fully understand the way MFU sets up formats, it may be helpful to describe the additional system variables Amstrad have inroduced to their BIOS to provide enhanced Disc Parameter information. This information takes the form of 10 bytes which are attached to the end of the standard CP/M DPB for each drive. Amstrad refer to the whole 27-byte data structure as an **eXtended Disc Parameter Block**, or **XDPB.**

The additional section is as shown below :

Bytes 0 to 16	Standard CP/M DPB
Byte 17	Sidedness - 0 = Single Sided; 1 = DS, Cylinder Access; 2 = DS, Side Access
Byte 18	Number of Tracks per Side
Byte 19	Number of Sectors per Track
Byte 20	First Sector Number
Bytes 21,22	Sector Size in Bytes
Byte 23	Gap length (R/W)
Byte 24	Gap length (Format)
Byte 25	FDC R/W mode (bitcoded MT/FM etc)
Byte 26	Auto-format detect flag - 0 = feature enabled; 1 = feature disabled

In general, it is **not** necessary to have a detailed understanding of these parameters. MFU enhances and alters these automatically as required; where relevant, they are discussed in later sections of this manual.

If you require more information on the structure of the XDPB for programming purposes, you should consult the appropriate Amstrad technical publications. Additionally, you will need an understanding of the programming and functions of the NEC uPD765 FDC chip; some discussion of this and sources of technical data are given in Appendix B.

D.4 Amstrad Disc Format Descriptor Block

All discs formatted with the Auto-detect Format option active on a PCW (or on a CPC under MFU) will have a Disc Format Descriptor Block written to Head 0, Track 0 ,Sector 1.

This data block contains a subset of the full XDPB, and is sufficient to allow the BIOS to construct an XDPB for that disc. This facility allows a PCW to use discs produced in a variety of formats and have each logged in correctly **without** the need to use MFU to set up a format before using such a disc.

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 contents of the Format Descriptor Block is as follows. 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 - As per XDPB spec.

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 (0Ah - 0Fh) : Reserved
(appear to be 00)

Appendix E

Disc Error Messages.

If MFU detects a disc error while analysing a disc or track, or reading or writing data, the operation in progress will be aborted and an appropriate error message displayed. This will be considerably more informative than CP/M's own high-level equivalents!

MFU receives error information direct from the computer's Disc Controller chip, and a knowledge of the operation of this chip is therefore an advantage if you need to be able to distinguish the subtle differences between some of the errors.

In addition to simply showing the appropriate error number and message, MFU 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.

The disc error codes translated by Version 4 of MFU are as follows. For compatibility, these are given the same Error Numbers as their BIOS equivalents, where these exist. BIOS errors are also explained.

Error Code	Message / Explanation
0	**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 it, or else you've forgotten to close the door lever. If neither of these is the case, then you may have a fault in the disc drive or its cabling - if the error recurs, then seek technical advice.
1	**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 a Write Protect sticker on it.

2	**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. The cause is usually that the track pitch of the disc is not the same as the FDC has been told! This is a "composite" error constructed by the BIOS and MFU from several low-level reports from the FDC chip. See also Error 5 (Missing Address Mark).
3	**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.
4	**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).
5	**Missing Address Mark** 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 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. However, MFU's Analysis Module has the ability to detect and read sectors marked in this way, so if in doubt investigate with A)nalyse. The analysis module can also write a DDAM'ed sector back to disc with a normal DAM if you wish.

6 **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 software or hardware failure has occurred.

7 **Spurious EOC fault - ignore**

This is a "phoney" FDC error code in the sense that it is not generated by the disc controller at all! It is generated internally by MFU. You should never see it on a PCW. If you get it on a CPC while running MFU, please let us know. It is used to cover, and correct for, a fault in Amstrad's CPC BIOS which generates spurious End Of Cylinder errors during certain FDC operations.

8 **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 XDPB. This will occur if you have put a different type of disc in Drive B and have tried to use it without first using MFU (or a stand-alone format setup program) to alter the XDPB format accordingly.

9 **Disc unsuitable**

This is another "fake" error code produced by Amstrad's BIOS. This can occur on a PCW8256/8512 if you try to write to, or format, a 40 track disc in Drive B while the Auto-Format detect facility is ON, and the PCW is set to believe that Drive B is an 80-track drive. MFU alters the BIOS to disable this "feature", so if you run MFU (you don't have to actually **do** anything), or run a stand-alone format setup program produced by MFU this problem will go away!

10 **Control Mark set**

This is another "fake" error code, used internally by MFU. You should **never** see it - if you do, please let us know! It is used to detect, and allow for, DDAMs.

255 **User Abort**

This one is fairly self-explanatory! Can occur if you abort from a very low-level operation such as a disc Sector Write.

Appendix F

Using Single Density discs on the PCW8256/8512

This Appendix is technical and describes hardware alterations to the PCW's main circuit board which allow the FDC to work in Single Density as well as normal Double Density mode.

The PCW8256 was not designed to be able to use **Single Density** FM-coded discs. However, the FDC and FDI (Floppy Disc Interface) ICs used by Amstrad do have this capability. To allow FM mode to to be used, some hardware modifications are required to the PCW's PCB. To be precise, two tracks require to be cut and two straps must be added between pins of two ICs.

There is unfortunately no equivalent simple modification which will allow a CPC6128 to use Single Density discs. CPC464 owners are however in the privileged position of not requiring any modifications - Amstrad's FDI-1 disc interface can use both Single and Double Density discs **without** alteration.

These modifications should ONLY be undertaken by a person with suitable experience of track lifting and strapping on "live" ICs, with the correct type of temperature-controlled soldering iron. Any person without the correct experience and/or equipment attempting this is likely to damage ICs and/or the PCB irreparably.

In any case, Moonstone Computing can accept NO liability whatsoever for the outcome of any attempt made to implement these modifications. The following information is provided ONLY as an aid to those able and willing to perform these modifications, on the understanding that they do so entirely at their own risk.

F.1 Removing the PCB

Unplug the PCW from the mains and leave for at least 30 minutes to allow high voltages in the tube circuitry to dissipate.

Place the PCW screen-down on a newspaper or blanket to prevent the screen being scratched. The rear case is held on by six cross-head screws; remove these and lift the rear case off carefully. Be careful not to snag any wires.

You will now see the single main circuit board beside the disc drives. **Carefully** remove the disc drive connector(s); the disc interface cable is soldered to the PCB and must be removed with it. NOTE its orientation BEFORE removal. Remove the small molex power connector(s) from the disc drive(s). Remove the large 5-pin power plug from the PSU and the two smaller molex plugs which supply cabling to the tube board and the keyboard.

The PCB should now be free. Remove it from the restraining clips. **Be careful** not to touch the ULA in the "hole" in the centre of the PCB. **This** is an SMD and its connections may be damaged if it is touched.

Observe all normal anti-static precautions while handling the PCB. **Place** conductive material such as aluminium foil on the bench and lay the board on top of this. Don't wear nylon materials or work in a room with such materials in the carpet. If in doubt, use an earth strap attached to your wrist. Your soldering iron MUST be earthed and temperature controlled as you will be working on the pins of "live" chips - **none** of the FDC components are socketed.

Inspect the layout of the circuitry on the board; familiarise yourself with the FDC/FDI ICs. The FDC is an NEC uPD765AC-2 chip, designated IC-203; the FDI is an SED9420-CAC, designated IC-201.

Turn the board over and find the positions of these ICs. Their pin numbers are marked on the mask in fives.

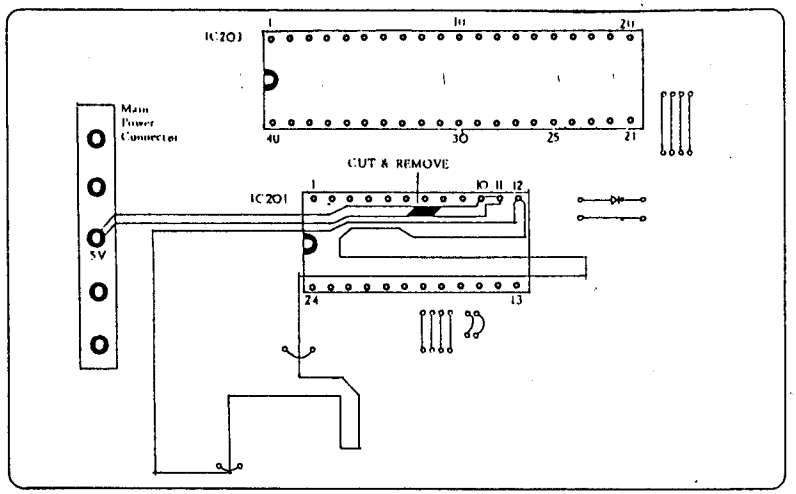
F.2 Technical Specification

The PCW's FDC/FDI circuitry cannot handle FM because - quite simply - Amstrad neglected to connect the FDC -FM/MFM signal (Pin 26) to the appropriate input (Pin 10) of the FDI chip.

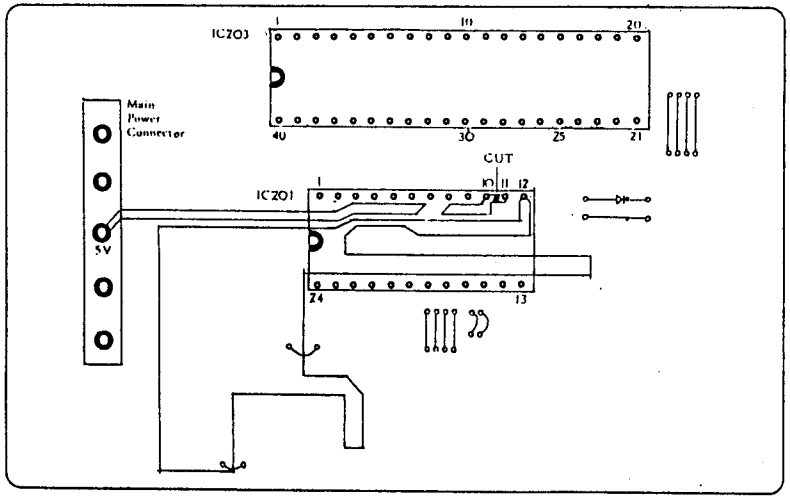
The FDI -FM/MFM pin is taken high on the PCB by direct connection to +5V supply. Unfortunately, this track is also common to the -STD/MINI input (Pin 11) of the FDI which must be kept strapped high. The inconvenient track topology on the PCB requires two track cuts and two straps to implement the above alteration.

F.3 Track Cuts

Cut and remove 3-5mm of track between **Main +5V supply connector** and **Pin 10 of IC-201** (the FDI) as shown below :



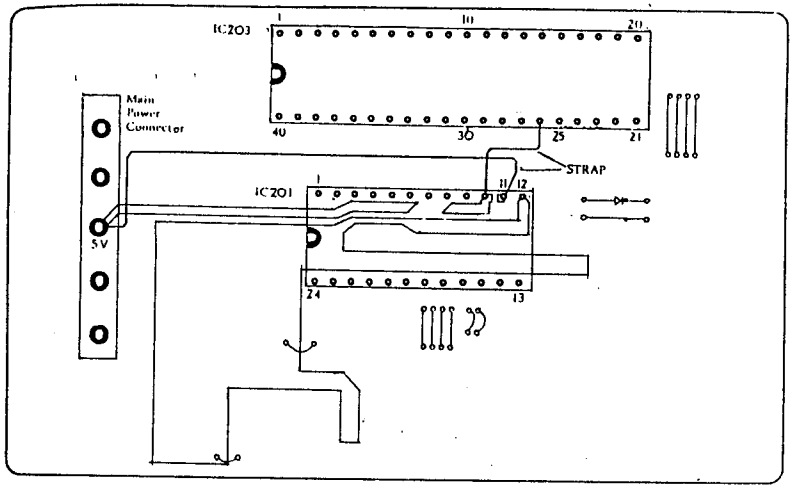
Cut the common track between adjacent **Pins 10 and 11** of **IC-201** (the FDI) as shown below. Be careful not to damage the thin track nearest to the IC or you'll have to strap this too!



F.4 Straps

Strap from **Pin 10** of **IC-201** (the FDI) to **Pin 26** of **IC-203** (the FDC) as shown below.

Strap from **Pin 11** of **IC-201** (the FDI) to **Main +5V Supply** as shown below.



F.5 Re-assembly and test

Test the straps for continuity with a multimeter; check the cut tracks for discontinuity. Fix straps to the PCB with correct adhesive or tape.

When you are satisfied that your work is complete, reassemble the PCB into the chassis. Reconnect the main Power plug. Reconnect the two minicons to mate correctly (they are colour-coded to match their sockets). Reconnect the power and interface cables to the drive(s). MAKE SURE that the ribbon cable to the drive(s) is connected correctly - there is NO keying.

Replace the rear cover and test the PCW. If the computer fails to boot, switch off and consider the symptoms.

1. **Drive Select LED and motors on continually; no seek or other activity.**

   Probably drive ribbon connector on wrong way round. Remove, reverse and retry.

2. **Drive restores (may then step) but no subsequent activity.**

   FDI/FDC data fault. Failure on data and/or synch. Check FDC for mistake - if strap soldered to Pin 24 instead of 26, you will lose Synch. If no apparent faults, check whole board for solder splashes, other tracks cut accidentally, dry joints etc. Note that NONE of the mods made should affect basic operation of the FDC/FDI APART FROM the strap from Pin 11 (IC-201) to +5V. Check the strap for continuity and shorts to ground; check that you have not strapped to the wrong power supply pin. If you have, you have probably killed the FDI.

3. **No activity whatsoever; no motors.**

   Fundamental fault. Check disc drive cabling and connectors. Check board thoroughly, using a meter where necessary, for solder splashes, tracks cut, incorrect strapping, especially to +5V, or physical damage. If no obvious reason, suspect damaged FDC and/or FDI. Replace FDI and retry. If fault persists, replace FDC also and retry.

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