Knife Plus

DISK RECOVERY SYSTEM

for Amstrad CP/M Plus Computers

HiSoft

High Quality Software

HiSoft Knife Plus

Complete Disk Recovery System

System Requirements:
Amstrad CPC6128, Amstrad PCW8256/8512, Amstrad PCW9512.

Knife Plus Manual, First Printing December 1987

Set using an Apple Macintosh™, Microsoft Word™ and Apple Laserwriter™.

Table of Contents

Welcome to Knife Plus 1

Back up your Knife Plus Master Disk 1
The Files on Your Disk 3

The Programs in Detail 5

The Knife Programs 5
The HiSoft Utility Programs 16

How to Recover a Disk 25

Introduction 25
Boot Sector Damaged 26
Directory Damaged 26
Other Tracks Damaged 27

Disk Structure 28

Sectors and tracks 28
CP/M Disks 30
Locoscript Disks 37

The Epilog 39

Bibliography 39
Technical Support 39
Other Programs from HiSoft 40

Welcome to Knife Plus

Knife Plus is a full-feature disk recovery system, allowing you to copy damaged disks, build up new versions of corrupted files and generally alter individual bytes anywhere on a disk.

The first section of this manual is devoted to the **Knife Plus** programs, explaining how they work and what they do.

The next section takes you through the process of recovering a damaged disk and many of you may want to read this section first, especially if you are not very happy using CP/M.

The final section explains the CP/M and Locoscript disk and directory structures and looks at the parts of this structure specific to the Amstrad implementation. This is a somewhat technical section and should be left to last.

The Epilog includes the Bibliography, details of technical support and other HiSoft products on the Amstrad Z80 computers.

First of all though, you should backup your master disk ...

Back up your Knife Plus Master Disk

PCW8256/8512/9512 Owners

We supply the **Knife Plus** master disk in a format known as *Old System Format*, as this is readable on all Amstrad CP/M computers.

Unfortunately, the DISCKIT utility supplied with the PCW machines is unable to copy disks in this format. To make a backup copy, you first must create a new blank disk. To do this, load up CP/M by turning off your computer, turn it on again and insert side 2 of your original system disk.

When CP/M has booted you will see an A> prompt at the left-hand edge of the screen; now type:

```
disckit [RETURN]
```

at the A> prompt. After a few seconds the program will be loaded; follow the instructions on the screen to format a disk. If you have a second disk drive note that, unless you really want to, there is no point in copying **Knife Plus** to a double-sided disk. Once you have formatted a blank disk, exit DISCKIT and put the **Knife Plus** master disk in drive A. Now type:

```
wp a: m: -q [RETURN]
```

This copies all the programs and files from your master disk onto the memory disk. When this has finished, and the A> prompt returns, put your newly-formatted disk in drive A and type:

```
m:wp m: a: -q [RETURN]
```

This process copies all the files on the memory disk onto your blank disk and therefore succeeds in backing-up your **Knife Plus** master.

CPC6128 owners

Load up CP/M Plus in the normal way by putting in your CP/M Plus master disk and typing `|CPM [RETURN]`. Now type:

```
DISCKIT3 [RETURN]
```

Follow the instructions to copy a disk, making sure that you copy from the **Knife Plus** master disk to a new, blank disk.

The Files on Your Disk

If you examine your disk, you should find the following files on it:

```
KNIFE+.COM
KMENU.COM
KNCONFIG.COM
SD.COM
WD.COM
WP.COM
PAGE.COM
GENSUB.COM
UNERA.COM
```

There may also be a file called `READ.ME`, as **Knife Plus** and the HiSoft utility programs are updated and improved from time to time; any recent changes will be described in this file which should be listed to the printer and stored with your manual.

The Programs in Detail

Please note that Digital Research/Amstrad licensing agreements do not allow us to provide the CP/M operating system on your **Knife Plus** master disk. This means that it is not possible to boot or warm boot CP/M from the disk. However, this will not affect you since CP/M Plus does not need a copy of the system on each disk.

The Knife Programs

If you are unsure about how CP/M or Locoscript treats disks then please skip this section and read the next section (entitled **Disk Structure**) first. If you want to read some general advice on recovering your disks before diving into **Knife Plus** then read the section after that entitled **Recovering Damaged Disks.** The following is the user guide on **Knife Plus.**

Start **Knife Plus** by booting CP/.M Plus from your CP/M system disk, then insert your **Knife Plus** working disk (the disk to which you copied your **Knife Plus** master) and then type:

```
kmenu [RETURN]
```

at the CP/M A> prompt. This program clears the screen, and prints up a menu that should look like:

```
HiSoft Knife Plus (vx.x) Menu
       PCW 8256/8512

    A     Use drive A
    B     Use drive B

    W     Copy whole disc

    K     Run Knife
    E     Exit to CP/M

   Enter a letter
```

Changing the Drive

Press A or B to select which drive you wish to work with. If you have only one drive, the Use drive B message will not appear and you will always edit the disk in drive A.

Sector Copying

The first thing you should do before attempting to recover a damaged disk is to sector copy it with the W option.

This option allows you to make a sector-by-sector copy of a disk in the selected drive. You should have both your corrupted disk (the disk you want to copy) and a blank (formatted or unformatted) disk ready.

Press W and the program will inform you how many swaps the copying process will take and then ask you to insert the source disk, which is the one you want to copy. If the boot sector (the very first sector on the disk) is damaged, you will be asked to remove the source disk and replace it with a disk with an undamaged boot sector. You can use any good disk of the right type (i.e. suitable for the relevant drive), then replace your source disk when asked.

The source disk will then be read; any bad sectors will be ignored (after trying to recover as much of them as possible). You will then be asked to put in your new, blank disk and the copier will copy the source disk data to it. Depending on the capacity of the disk and the RAM disk of your computer, you may then be told to put the source disk back in to repeat the process. With drive B disks this may take some time.

The big advantage that this copier has over DISCKIT is that DISCKIT will not ignore bad sectors, it simply gives up the copy. So, you cannot backup a damaged disk with DISCKIT.

The Knife+ Program

To edit the backup of your corrupted disk, you need to use the **Knife+** program. To run this, press K from the menu.

The **Knife+** program will be loaded and it will then print a sign-on message and ask you to select a drive. Press A to select drive A or B to select drive B. The program scans the disk in the specified drive, reads in the directory and then displays the first physical sector on the disk.

Amstrad disk physical sectors are 512 bytes long in all cases, but **Knife+** shows only 256 bytes at a time. The other half of the sector may be examined at any time by pressing a particular key (we'll tell you which one in a minute). The **Knife+** display looks like this:

```
CP/M Plus Knife Plus Disk Sector Editor V3.x (C) HiSoft 1987. PCW range.
 Drive: A    Track: 00    Sector: 01   Block: ****   File: -----------

0000 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
0010 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
0020 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
0030 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
0040 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
0050 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
0060 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
0070 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
0080 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
0090 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
00A0 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
00B0 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
00C0 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
00D0 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
00E0 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
00F0 HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH HH  AAAAAAAAAAAAAAAA
First Half   Press [ALT]-H for Help
```

There are 256 pairs of HH characters above and these refer to the 2-character hexadecimal representations of the 256 bytes being displayed. The 256 A characters above refer to the ASCII representations of these same bytes. There are 16 characters to each line.

The first four digits on each line represent the hexadecimal offset from the start of the sector of the first byte in the line.

The ASCII representations shown ignore the top bit of the character so the character having that value is displayed. Otherwise, a full-stop is printed.

The screen cursor will be flashing over the character representing the high nybble (4 bits, half a byte) of the first byte displayed. Any keys pressed by you will be interpreted by the program according to whether the key pressed forms a valid hexadecimal character or not; if it does, then its hexadecimal value is placed in the nybble under the cursor and the relevant parts of the display updated accordingly. If the key is a **Knife+** command key then that command is invoked. No alterations you make to the sector in memory will be written to disk until you issue either the write sector or paste sector commands. This means that you are free to experiment with data entry and commands until you feel confident. Remember, however, that any alterations you make are your responsibility - playing with disks at this level can easily be disastrous and HiSoft can accept no liability whatsoever for any loss or damage, consequential or otherwise, caused by the use of the programs on this disk.

The cursor may be moved about the display with the four cursor keys. Note that if you have programmed these keys using the CP/M Plus SETKEYS utility, most programs including **Knife+** will not necessarily be able to respond to them.

The second line of the **Knife+** display is known as its status line and shows the current drive letter along with the numbers of the track and sector currently being examined. These numbers are in hexadecimal and the track number starts from 00 while the sector number starts from 01. Following the sector number is the CP/M block number, also in hexadecimal. If the current sector is not part of a block (i.e. if it falls within the reserved tracks) then the block number is shown as ****. The terms *block* and *reserved track* are explained in the section entitled **Disk Structure** later in this manual.

The last part of the status line is the filename display. **Knife+** works out whether the current sector forms part of a file and if it does, the name of this file is shown. If the sector is not part of a file then the name is shown as ----------.

Knife+ Commands

Knife+ responds to a wide variety of commands, many of which are accessed by pressing the [CTRL] key the [SHIFT] key or the [EXTRA] key in conjunction with another. This means that the [CTRL] key (which is marked ALT on the PCW8256/8512/9512 machines), the [SHIFT] key or the [EXTRA] key must be held down while the other key is pressed.

Where the keys used differ between the CPC6128 **Knife+** program and the PCW8256/8512/9512 **Knife+** program, the PCW version is marked with a **§** sign.

Goto previous sector **[SHIFT]-Left arrow/[f7]§**

This command moves back to the previous sector and updates the display. If the current sector is the first on a track then the previous sector is the last on the previous track. If the current sector is the first on the disk then the previous sector is the last sector on the disk.

Goto next sector **[SHIFT]-Right arrow/[f1]§**

Advances to the next sector on the disk, updating the display as appropriate. If the current sector is the last on a track, then the next sector is the first sector on the next track. If the current sector is the last on the disk then the next sector is the first on the disk.

Goto previous track **[SHIFT]-Up arrow/boxed [-]**

This command moves to the previous track on the disk and updates the display. If the current track being edited is the first track on the disk then the previous track is the last track on the disk.

Goto next track **[SHIFT]-Down arrow/boxed[+]§**

Moves to the next track on the disk and updates the display. If the current track is the last track on the disk then the next track is the first track on the disk.

Select new track **[CTRL]-T/[ALT]-T§**

Prompts for a new track number, which may be entered in decimal or in hexadecimal, and then makes the current sector number on the new track the current sector. If the track number you give does not exist, you are re-prompted. The command may be aborted by pressing [ENTER] without entering a new number.

Select new sector **[CTRL]-S/[ALT]-S§**

Prompts for a new sector number, which may be entered in decimal or in hexadecimal, and then makes that sector the current sector. If the sector number you give does not exist, you are re-prompted. The command may be aborted by pressing [ENTER] without entering a new number.

Write sector **[CTRL]-W/[ALT]-W§**

This command causes the current sector contents in memory to be written back to disk at the location of the current sector. *Any changes you have made will be made to the disk, too, if you issue this command, so ensure that the data are correct before you do so!*

Paste sector **[CTRL]-Q/[ALT]-Q§**

Causes the current sector contents in memory to be written back to disk at a location you specify; it asks you for the new track and sector numbers and then writes the sector to that location. *Any changes you have made will be made to the disk, too, if you issue this command, so ensure that the data are correct before you do so!*

Search for a string **[CTRL]-F/[EXTRA]-F§**

Produces a prompt at the bottom of the screen, asking for a string to be searched for. Up to 80 characters may be typed here and the normal CP/M editing keys may be used while it is being entered. To abort the command, press [ENTER] without a string.

Once the string has been typed, press [ENTER] to start the search. All searches begin by displaying the current search mask, a value which is bitwise-ANDed with the string being searched for and the data being examined before any comparisons are made. This means that a search mask value of #FF (decimal 255) will cause a match to be found only if both strings are identical; a value of #DF will cause a match regardless of the case (lower- or upper-case) of any letters in the string. As #DF is often hard to remember, a value of 0 is regarded as being the same as #DF. Type in the new search mask value and press [ENTER], or type [ENTER] by itself to keep the value unchanged.

The search will begin immediately and the status line of the **Knife+** display is updated to show the track and sector numbers as they are examined. When the specified string is found, the sector in which it occurs is made the current sector and the display is updated to show it on the screen. The cursor is positioned at the first character of the string. If the string cannot be found by the time the last sector has been examined, the program prints an error message and, upon receiving a keypress, returns to the track and sector at which the search was started.

The search may be aborted at any time by pressing a key. When you do this, the sector which was under examination at the time becomes the current sector and the display is updated accordingly. Note that for a string to be found it must occur wholly within a physical sector, so if a string cannot be found on first search, it often is worthwhile trying again with a shorter portion of the same string

Search for a sequence of bytes **[CTRL]-G/[ALT]-G§**

This command is very similar to search for a string above except that the prompt asks for a sequence of bytes, each of which may be entered in hexadecimal or decimal. End the sequence by pressing [ENTER] alone to the Byte: prompt. The command may be aborted if it is mistakenly invoked by pressing [ENTER] alone in response to the first Byte: prompt.

Once the sequence has been typed the search proceeds in the same way as the string search; you will be asked for a new search mask value and the search then begins. Note that for a sequence to be found it must occur wholly within a physical sector, so if a sequence cannot be found on first search, it often is worthwhile trying again with a shorter portion of the same sequence.

Search for next occurrence **[CTRL]-N/[ALT]-N§**

This searches for the next occurrence of the last-entered string or byte-sequence. The search begins after the current character and the command allows a new search-mask value to be entered.

Show other half of sector **[COPY]/[EXTRA]-A§**

Updates the display to show the other 256-byte half of the current sector. If the current half is the first half, then the second half is shown. If the current half is the second half then the first half is shown.

Hexadecimal/ASCII switch **[TAB]**

Moves the cursor between the hexadecimal and ASCII representations of the sector's contents. When the cursor is over the hexadecimal representations, the data can be altered by typing in new hexadecimal values. When the cursor is over the ASCII representations, the data may be changed by typing in new ASCII characters. Command key values must be entered in hexadecimal, as **Knife+** invokes the command if the character itself is typed.

Quit edit **[ESC]/[STOP]§**

This command quits the current editing session and returns to the prompt asking you to select a drive. Any changes made to the sector in memory but not yet written to disk are lost. Any build-file that is open is automatically closed.

Select new block **[CTRL]-B/[ALT]-B§**

This command prompts for a new block number, which may be entered in decimal or in hexadecimal, and then makes the first sector of the new block the current sector. If the block number you give does not exist, you are re-prompted. The command may be aborted by pressing [ENTER] without entering a new number.

Go to next block in extent >

When tracking a file across a disk the ability to go to the next block in the file with a single keypress is extremely useful. This command allows you to do so within the confines of one extent (actually one directory entry but most Amstrad disks have only one extent per directory entry). If the current sector is not part of a block, is part of a block which is not occupied by a file or is part of the last block in an entry, the command is ignored. Otherwise the first sector of the next block is made the current sector and the display is updated.

Go to previous block in extent <

This command allows you to go to the previous block in a file within the confines of one extent (actually one directory entry but most Amstrad disks have only one extent per directory entry). If the current sector is not part of a block, is part of a block which is not occupied by a file or is part of the first block in an entry, the command is ignored. Otherwise the first sector of the previous block is made the current sector and the display is updated.

Go to start of named file [CTRL]-D/[ALT]-D§

This prompts you for a filename which should be entered in the normal way with up to 8 characters of primary filename and then, if there is an extension, the delimiting full-stop followed by up to 3 characters of extension. The letters may be entered in upper or lower case, or a mixture of both.

The directory is searched for this filename and if it is found in any user area then the program calculates the first block number holding this file and makes the first sector of that block the current sector. The display is updated accordingly. If the file cannot be found then an error message is displayed and you are re-prompted.

Filenames may be entered in short-form and **Knife+** will attempt a simple pattern-match. The short-form is simply as many characters of the name as you feel like entering. This part-name is searched for in the directory and is matched by the first filename which matches in the characters you have entered. For example, if your disk contains the files:

```
Knife.com
Knife+.com
Knife.gen
```

in that order in the directory, a part-name of `KNIFE` will find `KNIFE.COM`, one of `KNIFE+` will find `KNIFE+.COM` and `KNIFE.G` will match `KNIFE.GEN`. This does have the slightly disadvantageous side-effect of making `KNIFE.C` match `KNIFE.COM` rather than `KNIFE.C` if the former occurs first in the directory. To enter filenames with short extensions like this so that they will match with the correct files, pad out the extension the requisite number of spaces on the right.

Build Block — [CTRL]-Y/[EXTRA]-B

Add the current block into the Build File. If no Build File exists yet, you will be asked to give a name for it. The name should be up to 8 characters for the filename, then a full-stop followed by up to 3 characters for the extension e.g. TEMP.DAT.

A *Build File* is used to recover a corrupted file or a file that is missing from the directory. Using the **Build Block** and **Build Sector** commands you can re-create your missing file, chunk by chunk. This Build File must be created on another disk, not the source disk. On the PCW machines, the Build File is created on the memory disk, drive M and you should remember to copy it from there onto a real disk (using WP) before turning off your computer.

On the CPC6128 the Build File is created on drive B. If you do not have two drives then you will have to swap disks in drive A, you will be told when to do this.

While a file is being built, the Build File name will be shown on the bottom line of the screen. If there is no name there, no file is being built.

Build Sector — [CTRL]-X/[EXTRA]-R§

Add the current sector into the Build File. If no Build File exists yet, you will be asked to give a name for it. The name should be up to 8 characters for the filename, then a full-stop followed by up to 3 characters for the extension e.g. TEMP.DAT.

See the **Build Block** command for a discussion about Build Files.

Close Build File **[CTRL]-Z/[EXTRA]-C§**

Close the current Build File. Once closed, it cannot be re-opened without deleting it, so don't try! The Build File is automatically closed if you quit **Knife+**.

Show Help Screen **[CTRL]-H/[ALT]-H§**

Displays a help screen showing the various command keys and what they do. Hit any key to return to the main display.

How Knife+ knows the Disk Format

This is a technical section and you may omit it on first reading.

When you start an editing session of **Knife+** it needs to know various details about the disk to be edited. It finds this out by asking the *CP/M BIOS* (Basic Input/Output System) for the address of one of its data structures known as the *Disk Parameter Block* (DPB). The DPB holds vital facts such as the number of sectors a disk has, how big a CP/M block is, how many directory entries there are and so on. **Knife+** extracts the data it needs and uses it in many of the calculations it needs to perform to work out block numbers, filenames and so forth.

The BIOS used in Amstrad machines contains even more information than this, but **Knife+** doesn't use it directly. This allows the knowledgeable user to patch the DPB for a particular drive and make **Knife+** read a disk which has a strange format or which was not created on an Amstrad. For example, it is a very simple matter to alter the DPB to allow **Knife+** to read PCW disks on a CPC6128.

Details of how a DPB may be patched are not given here as it is no simple matter. If you are interested in this aspect of **Knife+**, you would need to be thoroughly familiar with CP/M, the BDOS and BIOS and the format of the disks you want to examine. Official Digital Research documentation is probably the best source of most of this information, or you can use The Amstrad CP/M Plus book (see Bibliography).

Regardless of the physical numbering of sectors, **Knife+** always displays sector numbers as beginning from 01.

Configuring Knife+

To enable you to alter the keys used to invoke commands in **Knife+**, we have provided a special program called KNCONFIG.COM which prompts you for new values and then alters **Knife+**. The program is run by typing its name at the CP/M prompt. As it resets the disk system there must be a disk in drive A while the program is running. The content of that disk is not important.

The program contains instructions for use which are displayed when it is run. It is fairly straightforward and enables you to alter the key values in a version of **Knife+** on any disk and under any name. The configured version is saved under the same name as the original, overwriting the contents. Configured versions of **Knife+** may be reconfigured at any time in the same manner.

Note that, if you do change the command keys, the Help screen may not be able to reflect these changes accurately.

The HiSoft Utility Programs

The programs supplied with **Knife Plus** are useful CP/M utilities designed to fulfil a range of purposes. Some will be useful to all users of disks and CP/M, others may be useful only to programmers. The important things about these programs from HiSoft's point of view are that they are small and easy-to-use. Each is described below; note that as each one is written to execute under CP/M, the files are stored on the disk as the name given plus the extension .COM.

An extremely powerful version of **Knife Plus**, called Knife86, is available for PC-DOS and MS-DOS machines which are IBM PC-compatible. This program can edit floppy and hard disks as well as non-DOS disks and knows all about DOS 2 features such as sub-directories and clusters. Please call HiSoft for further details.

WD

This program allows you selectively to delete files on a disk. **WD** is invoked by typing its name followed by a standard CP/M ambiguous file specification such as B:*.COM. If no ambiguous file specification is given, **WD** treats it as if *.* has been entered. If a drive specification only is entered, such as B:, **WD** treats it as *.* on the specified drive.

The name of each file which is found to match the file specification is then displayed along with a prompt

```
Delete (Y/N/A/Q/P)?
```

If you hit the Y or y key, the displayed file is deleted. Pressing N or n will not delete this file and skip to the next one. If you press the A or a key, this and all subsequent files which match the specification are deleted without further prompting. If you press Q or q, the program ends and returns to CP/M without deleting the current or subsequent files. Pressing P or p takes you back to the previous file, if there is one, and gives you the opportunity of deleting it. If you press any other key, the file is not deleted and the name of the next matching file is displayed.

If any file which is set to Read Only or is protected in some other way is selected for deletion with **WD**, the program prints the message Cannot delete! and continues to the next file.

SD

This utility produces and displays a detailed listing of the files on a disk Like DIR, **SD** may be invoked by itself or it may be followed by a CP/M ambiguous file specification such as `A:KNIFE+.*`. For example, to get a complete listing, type

```
sd [ENTER]
```

or

```
sd a: [ENTER]
```

and so on.

To get a listing of all files with the .COM extension, type

```
sd *.com [ENTER]
```

or

```
sd b:*.com [ENTER]
```

etc.

The listing produced on the screen is made to fit into 80 columns and consists of up to two filenames per screen line. Each name is shown in full, followed by an S if the file's system bit is set and followed by an R if the file's read-only bit is set. A file may have both attribute bits set, in which case both S and R are shown. Please see earlier sections of this manual for a complete description of system and read-only attribute bits.

These one-letter flags are followed by the length of the file in K (1024-byte units), rounded up to the nearest 1K. This number is displayed in decimal. It is followed by the letter K to make its meaning clear. After this comes the number of records in the file. A record is a 128-byte unit used by CP/M as the basis of all its disk operations. A file always occupies an integral number of records whatever its physical length. The number displayed is in decimal. When all the filenames which match the ambiguous file specification have been displayed, **SD** prints the amount of disk space free in 1K units.

SD has the advantage over CP/M utilities such as STAT, SHOW and DIR (FULL) in that it is less than 1K long; the other utilities are between 6K and 15K long. **SD** does not sort the filenames into alphabetical order.

WP

WP is a short-and-sweet file-copying utility which makes the task of transferring given files between disks relatively easy. It is used like this:

```
wp <afn> {<destination drive name>} {-q} [ENTER]
```

(The curly brackets, { and }, specify an optional item) where <afn> is a standard CP/M ambiguous file specification from a lone drive name such as A: up to a complete filename such as A:KNIFE+.COM. Being an ambiguous file specification, the standard CP/M wildcard characters, ? and *, may be used to match a group of files.

Note that a lone drive name is taken as an ambiguous file specification which matches all files on the specified disk in the current user area. The <destination drive name>, if present, is a normal CP/M drive name such as B:. If left out, the default drive is used. The -q is the quiet option switch and its action will be described shortly.

WP finds each file matching the <afn> given and presents the name of each one to the user along with a prompt:

```
Copy (Y/N/A/Q/P/B/W)?
```

Y corresponds to *Yes*, N to *No*, A to *this and all subsequent matching files*, Q to *Quit*, P to *goto previous selection*, if there is one, B for *create a backup* (with an extension of .BAK) of the file when copied and W stands for *without backup* (useful if you have used the B option on the command line and don't want a backup of this particular file).

Each file to be copied is copied to the destination drive with the same name as it has on the source drive. Any file of the same name is deleted prior to the operation whether the copy succeeds or not, unless it specifically has been marked read-only.

If the -q option is present (the q may be in upper or lower case) in the command line then the Copy (Y/N/A/Q/P/B/W)? prompt will not appear. **WP** will run as if A had been pressed in response to this prompt before nay file names were displayed. In other words all files matching the <afn> are copied without prompting. Note that the -q option must be after the <afn> and the <destination drive name> (if present) in the command line.

If the source and destination drives are the same an error message is printed and no names are produced. If the copy fails for lack of disk or directory space a message is printed to the console and the new copy of the file is deleted.

PAGE

PAGE is an extremely useful program which prints out a number of files to the program's standard output. It is usual to re-direct this output to the printer or to a file to be spooled at a later date. Output may be redirected in the standard Unix / MS-DOS manner by specifying a filename preceded by the > character in the command line. The named file (or device) is opened and all PAGE's standard output is sent to the file rather than to the screen.

When PAGE terminates, the file is closed. Note that sign-on messages and file status reports are sent by PAGE to its standard error, which is not re-directable and appears always on the screen. The filename used to redirect standard output (which also is the screen until told otherwise) may be a CP/M device such as LST: (the printer) and PUN: (the serial port).

The most useful features of PAGE are that each of the important paper characteristics may be altered and that command lines may include wildcards (ambiguous filenames) to specify a group or groups of files.

The general form of a PAGE command line is

```
page <afn> {<afn> <afn> etc} {<options>} [ENTER]
```

If page by itself is typed at the CP/M prompt the usage message is produced. This is a very brief summary of the command-line syntax and the options available. Each option may be placed anywhere in the command line and each one affects all files. An option is introduced with the - character, which is followed immediately by the single-letter option. Six of the option letters take a decimal number as a parameter and this must be placed immediately following the option letter. The options available are:

Bnn Set bottom margin to nn

D print the date and time at the right hand edge of the title

Enn set tab size to nn (the default is 8)

F Don't use formfeeds

Hnn set the header margin to nn lines

Lnn Set line length to nn

N No output; show matching filenames only

NO print line numbers (from 1) at the start of each line

Pnn Set page length to nn lines

Q Quiet; don't output status messages, only errors

R don't double-strike filenames

S Single-sheet printing; pause after each page

Tnn Set top margin to nn

The N option is useful when you want to see what files you have specified before paging them. The S option is useful when sending output to a sheet-feed printer; the program pauses after each page has been output and waits for a key to be pressed before proceeding to the next page.

A typical **PAGE** command line would be:

```
page -sfp50 *.c myfile.asm print.me! >lst: [ENTER]
```

This causes all files with the extension .C to be paged along with MYFILE.ASM and PRINT.ME!; the output is sent to the printer (>lst:) and the page-length is set to fifty lines. The requisite number of linefeeds, rather than formfeeds, are used to separate pages (f) and the program pauses after each page (s).

PAGE sends each file to the standard output in pages, with the name of the current file printed in the top left-hand corner and the current page number in the top right-hand corner. The filename is over-printed by sending it to the printer followed by a lone carriage-return and then sending the name again. On almost all printers, this causes the name to be over-printed, giving an emboldened effect.

The default page-length is 66 lines, the line length is 80 characters and the top and bottom margins are 3 lines each. Formfeeds are used to separate pages of output but this may be altered with the -F option.

GENSUB

This program is used to generate batch (submit) files. It generates a list of filenames matching a given or defaulted CP/M ambiguous file specification (afn) and precedes each one with a given command. Optionally a list of parameters may be appended to each line, and **GENSUB** is able to recognise certain special characters and perform specific actions on meeting these. The general form of a **GENSUB** command line is:

```
gensub {-e} {<afn> {<first command> {<command tail>} } } [ENTER]
```

where items enclosed in curly brackets (braces) are optional. The -e option, if present, must be the first item on the command line. All items in the command line have default values but if an item in the list is included, all preceding items above also must be included. The -e option is an exception to this rule - it may be left out if not required.

Each line of output produced by **GENSUB** is sent to its standard output. This is usually the screen but may be re-directed in the standard Unix / MS-DOS manner by specifying the file or device name to receive the output preceded by a > character in the command line. See the description of **PAGE** above for more details on redirection.

The output produced by **GENSUB** is formed as follows:

First, the item specified by <first command> is output Then, the current filename is output. Finally, the items specified by <command tail> are output

If <command tail> is not present, it defaults to no items; if <first command> is not present, it defaults to WP. The <afn> parameter, if not specified, defaults to *.*. This parameter may consist only a drive name, such as M:, in which case it matches all files on the specified drive in the current user area.

The <command tail> item, as it is the last item on **GENSUB**'s command line, can be an arbitrarily-long list of items. <first command>, however, is determined by its position in the command line and thus can comprise only one item. To get around this limitation, <first command> may be a list of items enclosed wholly in double quotes:

```
"command1 -aflag another theend"
```

GENSUB offers a limited amount of macro expansion. If the dollar sign, $, is found anywhere in <first command> or <command tail> it is treated specially by the program.

The dollar sign itself is discarded and the character immediately following the dollar sign is interpreted in the following way:

@ is expanded to the full filename of the current file

* is expanded to the name only (no extension) of the file

_ is expanded to a newline sequence

$ is used to represent the dollar sign itself

So, if **GENSUB** was invoked with this command line:

```
gensub *.c "cc -n" $_as -zap $*$_ln $* -lmylib -ls -lc >doit.sub
```

and the disk contained the following .C files

first.c, myprog.c, gensub.c

then when **GENSUB** finished the file DOIT.SUB would contain the following lines:

```
CC -N FIRST.C
AS -ZAP FIRST
LN FIRST -LMYLIB -LS -LC
CC -N MYPROG.C
AS -ZAP MYPROG
LN MYPROG -LMYLIB -LS -LC
CC -N GENSUB.C
AS -ZAP GENSUB
LN GENSUB -LMYLIB -lS -LC
```

which could then be executed using the Digital Research SUBMIT utility. Neither <first command> nor <command tail> may exceed 1023 characters after expansion, although no checks are made by the program to see if this is adhered to. As SUBMIT itself is able to handle lines no longer than 127 characters, this should not be a problem.

UNERA

UNERA is an extremely useful utility which can unerase a file which has just been removed with the **ERA** or **WD** commands. It can work only if no other alterations have been made to the disk or directory. In other words, if you have edited the disk with **Knife+**, if you have subsequently saved another file or altered the directory in any way, **UNERA** will probably not work!

It is most useful immediately after the injudicious use of **ERA** or **WD**. Simply type

```
unera <filename> [ENTER]
```

where <filename> is the name of the file you want un-deleted. The program will try its best to recover the file but please remember the restrictions and warnings given above. Good luck!

How to Recover a Disk

Introduction

Firstly, you should always make a sector copy of the damaged disk. Do this by formatting a new disk using DISCKIT on the PCW computers or DISCKIT3 on the CPC6128. Format the disk to be the same type as your damaged disk.

If you have a PCW computer, it will be useful to copy the **Knife Plus** programs to the M: drive by inserting your backup of the **Knife Plus** disk and typing:

```
WP KMENU.COM M: -Q [RETURN]
```

and, when this has finished:

```
WP KNIFE+.COM M -Q: [RETURN]
```

and now invoke the menu by typing:

```
M: [RETURN]
KMENU [RETURN]
```

If you have a CPC6128, insert your backup of **Knife Plus** in your A drive and type:

```
KMENU [RETURN]
```

In both cases, the menu should now be on the screen; choose which drive you want to copy on. This will depend on which type of disk is corrupted, a single-sided type (PCW8256/8512 drive A or CPC6128 drive A and B) or a double-sided, higher-capacity type (drive B of the PCW8512 and drive A of the PCW9512). Choose A or B and insert your corrupted disk in the relevant drive.

Now type W to copy this disk onto your newly-formatted disk; you will be asked to change disks every so often and this may take quite a long time for double-sided disks but it will be worth it.

If the boot sector of your source disk is damaged then the sector copier will ask you to insert a good disk of the same type as your source disk so that the copier can log-in the disk as the correct type. Just use any disk that has a good boot sector and is of the same type (drive A or drive B) as your corrupted disk. After the copier has logged-in this disk, it will ask you to replace your damaged source disk so that the copy can proceed.

Eventually, the menu will re-appear. What you do now will depend on in what way your disk was corrupted.

Boot Sector Damaged

If you had an error which claimed that Track 0 was damaged and that was the only problem, then the sector copy will have cured this for you and you need take no further action, simply exit the menu by typing E and get on with the washing-up or something else equally constructive!

Directory Damaged

The disk directory is on Track 2 for single-sided discs (the ones that go in drive A of the PCW8256/8512 or any drive of the CPC6128) and on Track 1 for double-sided (PCW8512 drive B and PCW9512 drive A) disks.

If you had an error report blaming the directory track, you've got problems and it is going to take some time to recover your files.

Once the directory has been corrupted you often cannot tell where your files are on the disc anymore since it is the directory that holds this information in the shape of a set of block numbers for each file; see the next section for more detail.

If these filename entries, along with the block numbers, have been damaged, then the only thing you can do is search the new disc (using the search commands of **Knife+**) to try and find where your file is. Note down the block numbers of any data you find and try to work out all the blocks that make up the file, in order.

Then use the build-a-file commands to create the file again; the file will be built on the RAM-disk, M, of the PCW machines or on drive B of the CPC6128. Once you have built up the file, or as much of it that you can find, then exit **Knife+** (press [STOP] on the PCW or [ESC] on the CPC6128) and then the menu (type E) and copy the built-file from M or B to your new disk using our **WP** utility or PIP.

You can do this for all the files you have lost. Obviously it may take some time!

Other Tracks Damaged

If other tracks have been damaged, then you may have lost some of the data of your files. You should be able to find this out by looking at the directory, going to the blocks of your file, one by one, and looking to see if the data makes any sense.

If you have lost part of the information in a file, you can either choose to build the file again (as described above) or edit the data on the disk directly by typing in new data and then saving the sector with [CTRL]-W/[ALT]-W.

We hope that the above has given you some insight in how to recover your damaged disks; the most important thing to remember is ***Don't Panic***, just take it slowly and carefully and, in most cases, you should be able to recover a substantial part of your lost information.

Disk Structure

Now that you have your copy of **Knife Plus**, you can begin to hack away at disks to your heart's content. However, for the program to be really useful a full understanding of the way CP/M and Locoscript organises files on the disk is most important.

Every disk formatted on a CP/M system has a similar structure, and the Amstrad computers' disk system is no exception. Once certain basic disk parameters have been discovered, the organisation of the disk can be relied upon to be thoroughly predictable. The same applies to Locoscript disks which have an almost identical structure to those of CP/M.

Before we look at how we can determine and use these parameters, we must understand the concepts involved in floppy disk technology as a whole. This means we must know what is meant by *sectors* and *tracks*.

Sectors and tracks

A floppy disk is essentially a piece of plastic covered in a magnetic film which is very similar to that found on cassette and video tapes. A tape system has the disadvantage that, in order to read or write a particular piece of data, the length of tape preceding the required area has to be traversed by the read/write head before the data can be acquired. Disk drives get over this problem by moving the head as well as the magnetic film. The disk itself is spinning at a very high speed and the head is able to move along a radius of the disk, covering any given annulus at any given time. Obviously, a motor is required to move the head across the disk's surface.

We can control a motor extremely precisely if we allow it to move in pre-defined steps, so it follows that if each step corresponds to a part of the disk which contains (or is to contain) data, the data can be recorded and retrieved with great speed and considerable accuracy. To facilitate this process, the data is stored in a given number of circular *tracks* on the disk.

The number of tracks per disk is fixed at the time of manufacture; the two most common standards are to have 40 tracks or 80 tracks. In the case of a 40-track disk, such as used by most of the Amstrad disk formats, this means that the entire working area of the disk is divided into forty annular regions, to each of which the read/write head may be moved.

The amount of data which may be held in a track is often very substantial and as a result most machines are unable to handle it all in one go. To solve this problem each track is divided up into divisions called *sectors*. The number of sectors within a track is a function of the system's hardware and software and is one of the disk parameters we mentioned earlier. Likewise, the amount of data which may be held in a sector is determined by another parameter. This is known as the sector size; common sector sizes are 128, 256, 512 and 1024 bytes - the Amstrad disks always have 512-byte sectors.

In most cases the Amstrad's disks are formatted to have 9 sectors of 512 bytes each per track and as there are 40 tracks it follows that a side of a disk may hold up to 40 * (9 * 512) bytes which works out as 184320 bytes, or 180K. When a disk is formatted, it sets up the sectors and marks the sectors within a track with a number. The first sector in a track is usually numbered 0 or 1, but Amstrad has chosen to use strange values such as 65 (#41) and 193 (#C1) for some of its disk formats.

With our newly-gained information on tracks and sectors, we can see that it is possible to refer to some particular data by its track number and by its sector number. Track numbers always start at 0, so the very first sector on a single-sided Amstrad disk is

Track 0 Sector 01

and the last sector on such a disk is

Track 39 Sector 09

The disks known as CF2-DD, which go into drive B on the PCW8-series machines (or drive A of the PCW9512), have 160 tracks, numbered from 0 to 159.

The ability to refer uniquely to a sector is surprisingly useful, as we are about to find out, but first a warning:

The CP/M operating system believes that sectors always are 128 bytes long. These mythical sectors are also known as records, logical sectors or CP/M sectors. Throughout the rest of this guide we call them *records*. However, when **Knife Plus** displays a sector number it is talking about a real sector and the display refers always to physical 512-byte sectors rather than CP/M's logical sectors.

CP/M Disks

The CP/M operating system deals with data in groups of 128 bytes regardless of the physical configuration of the sectors. The 128-byte groups are known as records in CP/M terminology and happen to be the smallest amount of data which the operating system acts on with reference to disks. This means that every CP/M disk is an exact multiple of 128 bytes in length. Now read again the large print warning above! The next step up in data handling is the block, which is a group of records belonging to the same file. A block is usually one of 1024, 2048 or 4096 bytes long, which means that there will be either 8, 16 or 32 records per block.

All Amstrad disk systems except the PCW CF2-DD disks have 8 records (= 1024 bytes) per block, which means that a block occupies 2 physical sectors. CF2-DD disks have 16 records per block so that a block occupies 4 physical sectors. In both cases the sectors will be contiguous in numbering terms, so if the first block occupies sectors 01 and 02 of track 2 the second block will occupy sectors 03 and 04 of the same track. Files always occupy a whole number of blocks, even if the physical length of a file would seem to dictate otherwise. This means that a file which is five records long will occupy exactly the same amount of disk space as one which is seven records long. It is clear that the smaller the block size, the more efficient the use of disk space, but at the same time a small block size reduces the speed with which a file may be read or written. A compromise is arrived at and as floppy disks are now rather cheap, a large block size is often adopted. The directory of a CP/M disk always occupies block 0 and possibly other consecutive blocks as well. The directory is the place in which the system keeps a record of all the files on the disk and where they can be found. Once we've found the first directory block, the rest is easy. But how do we calculate the location of the directory? This is where the disk parameters are required.

The information we need is shown below for each of the Amstrad formats. Remember that not all Amstrad machines support all the disk formats so if there is one you do not recognise, don't worry!

Amstrad IBM Format

Single-sided
512-byte physical sector size
40 tracks numbered 0 to 39
1024-byte block size
64 directory entries
8 sectors per track numbered #01 to #08
1 reserved track

Amstrad System / Vendor Format (CPC6128)

Single-sided
512-byte physical sector size
40 tracks numbered 0 to 39
1024-byte block size
64 directory entries
9 sectors per track numbered #41 to #49
2 reserved tracks

Amstrad Data Format (CPC6128)

Single-sided
512-byte physical sector size
40 tracks numbered 0 to 39
1024-byte block size
64 directory entries
9 sectors per track numbered #C1 to #C9
0 reserved tracks

Amstrad CF2 Format (PCW8256/PCW8512 drive A format)

Single-sided
512-byte physical sector size
40 tracks numbered 0 to 39
1024-byte block size
64 directory entries
9 sectors per track numbered #01 to #09
2 reserved tracks

Amstrad CF2-DD Format (PCW8256/8512 drive B format/PCW9512 drive A format)

Double-sided
512-byte physical sector size
160 tracks numbered 0 to 159
2048-byte block size
256 directory entries
9 sectors per track numbered #01 to #09
1 reserved track

Some of this information is not new to us, but the part which gets us to the directory is is the number of reserved tracks. A CP/M disk almost always has some reserved tracks. These tracks may be used to hold the code required to boot the system into the machine and also may house the CP/M BDOS (Basic Disk Operating System) and the CCP (Console Command Processor). Reserved tracks always start from the beginning of the disk so if the format says *2 reserved tracks* it means that tracks 0 and 1 are used by the system for its own purposes.

The directory follows the reserved tracks, so the first directory record of a disk with 2 reserved tracks must be

```
Track 2 Record 0
```

Let's examine the first few bytes of a typical directory to find out how these things are laid out:

```
0000 00 50 49 50 20 20 20 20 20 43 4F 4D 00 00 00 3A .PIP     COM...:
0010 02 03 04 05 06 07 08 09 00 00 00 00 00 00 00 00 ................
0020 E5 43 4F 55 4E 54 20 20 20 47 45 4E 00 00 00 14 .COUNT   GEN....
0030 0A 0B 0C 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

It's obvious from this that each directory entry occupies 32 bytes, which enables us to gain yet more information about the disk structure. If we take this entry as being part of an Amstrad CPC system format disk you'll remember that the disk parameters mentioned that there are 64 directory entries. If each directory entry is 32 bytes long, the entire amount of disk space devoted to the directory must be 32 * 64 bytes, which is 2048 bytes or 2 blocks. This means in turn that the directory on a system format disk occupies sectors 0-7 on track 2.

So now we know where the directory starts and how long it is. As it occupies two blocks on an Amstrad CPC system format disk, it follows that the first block free for data is block 2. Let's take a look at an individual directory entry and see what information we can gain from this:

```
0000 00 50 49 50 20 20 20 20 20 43 4F 4D 00 00 00 3A .PIP     COM...:
0010 02 03 04 05 06 07 08 09 00 00 00 00 00 00 00 00 ................
```

The very first byte of the entry, which is 00 here, tells us the status of the file. If this byte is 0, as above, it means that the file exists. If it is #E5, the file has been deleted. A deleted file's directory entry can be overwritten by the next file to be created on the disk and all files whose first directory byte is #E5 will not be found in a CP/M BDOS open call. As well as 0 or #E5, the first byte may also be any number from 1 to 15. This means that the file exists, but in a different user area. Normally when using CP/M we are in user area 0 but we can issue the built-in USER command to move to another user area. The main use of this is on large (hard) disks where directories can get too large to comprehend unless they are split into different areas.You can also hide files by putting them into a different user area.

The next eight bytes of the entry comprise the file name. The characters are always stored by the system as upper case and the field is right-filled with spaces (CHR$(32)). The three bytes following this comprise the file extension, which is again in upper case and right-filled with spaces. If you make any alterations to the directory entry, remember to use only valid characters in the filename and extension fields and use only upper case letters as otherwise you may not be able to access the file!

The full-stop which we use to separate filenames and extensions, as in

```
GEN80 MYFILE.GEN
```

is not stored in the directory; there is no need as the name and extension fields are always eight and three bytes long respectively.

To simplify the next part of the discussion, we're going to ignore something called an extent until a little later. The sixteenth byte of a directory entry, which is #3A in our example above, tells us how many records there are in the file. We can see that PIP.COM has #3A records, which is 58 in decimal. As there are 8 records per block, it follows that PIP.COM will occupy 8 blocks, but the first two records only in the final block form part of the file. So far so good, but how do we know where to start looking on the disk for PIP.COM?

The answer lies in the next few bytes. These bytes, from byte 17 in a directory entry onwards, are the numbers of the blocks occupied by the file, in the order in which they were written. All Amstrad disk formats except the PCW CF2-DD disks use block numbers between #00 and #FF, which means that the numbers fit into a byte. These are referred to as 8-bit block numbers. The CF2-DD disk format has more than #FF blocks, so the numbers have to be stored in 16 bits. Hence, they are referred to as 16-bit block numbers.

We can see that PIP.COM occupies blocks 2, 3, 4, 5, 6, 7, 8 and 9. We also know that all 8 records in blocks 2 to 8 are part of PIP, but only the first 2 records in block 9. Block 0 starts at track 2 record 0, block 1 at track 2 record 8, and block 2 at track 2 record 16, which is the first record containing part of PIP. It follows through sequentially like this, so it's dead easy to find every single record of a file. There are times when a file will not occupy consecutive blocks as other, smaller, files may have been deleted and the space re-used by the new file. Nevertheless, it is not at all difficult to translate block numbers from a directory entry into physical sectors on a disk, and thus locate every record of a file. **Knife+** even does this for you. There is one problem, though...

Extents

As the last 16 bytes of a directory entry are used to hold the block numbers occupied by the file, it follows that each directory entry may reference only 16 blocks, which in the case of the Amstrad means 16K. This is a bit of a limitation as we are sure to want to read and write files which are longer than 16K from time to time. CP/M gets over this by introducing the concept of an extent, each of which occupies one directory entry and can therefore be no longer than 16K. If a file is too long to be held in one extent, the CP/M BDOS opens another extent for it, creating a directory entry identical to the previous one except that the new entry reflects only the number of records in this extent and the block numbers refer to this extent only. If the file needs yet another extent, the process continues, and so on.

Each directory entry, although referring to a different part of the file, has the same filename entry, so how do we tell which directory entry refers to which extent? You guessed it! One of the spare directory entry bytes is used to tell us which extent this entry refers to. The byte in question is byte 13, which is the first byte following the last character of the filename extension. The first extent of a file has this byte set to 0, the second to 1 and so on. For the same reason that individual parts of a file may not necessarily be saved on contiguous blocks, so extents may not necessarily appear in order in the directory. It all depends on where the free space is. If, when you examine a file's directory entry on an Amstrad CP/M or Amsdos disk, you find that it doesn't refer to the first extent, keep on looking. Likewise, when you read an entry and find that there are 128 (#80) records in that file, check to see if there are any more extents recorded in the directory. There may not be, as there is nothing to stop a file being an exact multiple of 16K bytes long, but you must check to make sure.

A problem with extents is that high-capacity disks with large block sizes may store more than more than one extent in each directory entry. This is shown by having a field in the DPB (Disk Parameter Block) set to the number of extents per directory entry minus 1. This field is known as EXM (the extent mask) and it usually is therefore 0. If the EXM value is non-zero, there is more than one extent per entry, and the byte in a directory entry referring to the number of records occupied is valid only for the highest-numbered extent in this entry; all lower-numbered extents must by definition have #80 records.

The final pieces of information held in a directory entry are cleverly hidden away in the 3-byte filename extension field. These bytes are normally capital letters or other ASCII characters, which means they are below #80 (128) in value. This in turn means that the very top bit, bit 7, is zero in all cases. As CP/M knows this, it takes advantage of the fact and uses this bit to store vital pieces of information. The top bit of the first character of the extension tells the system whether the file is set to R/W (read/write) or R/O (read-only) access. If the bit is clear (zero), as it is normally, the file is considered to be a read/write file, while if it is set (1), the file is read-only. The top bit of the second character of the extension is used to signify whether the file is a directory (DIR) file or a system (SYS) file. A '1' corresponds to SYS and a '0' to DIR. A file marked as SYS will not be listed in normal directory listings.

Locoscript Disks

Locoscript is the built-in word processor available on the PCW computers; if you have a CPC6128 you need not read this section.

Locoscript handles disks in a way that is remarkably similar to that of CP/M. It maintains a directory in the same place (i.e. track 1 for CF2-DD disks or track 2 for CF2 single-sided disks) and the directory is almost identical to CP/M's.

The are three things to watch out for when recovering Locoscript documents:

1. Locoscript *Groups* are analogous to CP/M's *User numbers* i.e. if a document is in group 1, the first character of its directory entry will be 01 instead of 00, if it is in group 7, this first character will be 07 etc.

2. Limbo files are files which are temporarily erased but can be recovered. These are marked in the directory by Locoscript by adding 8 to the group number in the first character of the document's directory entry e.g.

```
0020 0D 54 45 53 54 20 20 20 20 54 58 54 00 00 00 04 .TEST    TXT....
0030 30 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

   If TEST.TXT is a Locoscript file, it is in limbo and in group 5 since the first character in the directory entry is 0D (hexadecimal) which is 13 in decimal which is 8 (limbo) plus 5 (group).

3. When trying to re-build a lost Locoscript document it is very important to ascertain where the beginning and the end of the document are; otherwise Locoscript will become very confused when trying to read the recovered document.

The beginning of every Locoscript document contains some header information which appears to specify the template being used *inter alia* Therefore, if you have found the beginning of the text of your document, you should always go back to the start of the block in which you found this text to check to see that it has some information about the document's template (i.e. the name of the template). When you have ascertained this you can start building the file, beginning with the start of this block.

It is also most important to get the length of the document right; if the document is only 1 sector long (which is the minimum it can be) then you must not build the whole block (using `[EXTRA]-B`) but only the sector (`[EXTRA]-R`). Otherwise, Locoscript will try to make sense of all the garbage off the end of the document and get very confused. So you should always check through the block you are thinking of saving (use `[f1]` to skip to the next sector, `[EXTRA]-A` to show other half of sector and `[BOXED]-PLUS` to get the next track in the block) to see if it contains the end of the document.

If it does, work out how many sectors of this block are used by the document (remember, a sector is 512 bytes and a block is either 1024 bytes (for CF2 single-sided disks) or 2048 bytes (for CF2-DD double-sided disks)) and then build only that number of sectors into your build file; do this by using `[EXTRA]-R`, then `[f1]`, then `[EXTRA]-R` etc.

The above information should help you to recover lost Locoscript documents but please be sure to read through the above again together with the section on **CP/M Disks** and the section on **How to Recover a Disk** before trying to do it yourself.

The Epilog

Bibliography

The following books are excellent for learning about the way CP/M handles its disk system; the last one being especially useful for the Amstrad systems.

CP/M User Guide	Digital Research	Tel: (0635) 35304
CP/M Plus Handbook	Digital Research	Tel: (0635) 35304
The Amstrad CP/M Plus	MML Systems	Tel: (01) 247 0691§

§ Available from HiSoft in paperback at £10.95.

Technical Support

If you experience any problems with **Knife Plus** then please feel free to write to us or telephone between 3pm and 4pm every weekday.

Please have the following to hand when you call us and include them if you are writing to us:

1. Your name!
2. The serial number that is on your master disk.
3. The name of the product you are using (**Knife Plus**) and its version number (this is displayed at the top of every edit screen).
4. The type of computer you are using, how many drives it has, its memory etc.
5. A daytime phone number so we can get back to you if need be.

If you think you have found a bug, try to make it reproducible so that we can see for ourselves that it is real.

It is often easier for us to answer by letter, especially if you think you have found an obscure bug. However, you should always allow 2-3 weeks for a written reply and if you require urgent action it may well be better to phone in the technical hour (3-4pm) to see if the problem is already known.

Other Programs from HiSoft

Many other language and utility programs are available for your Amstrad computer from HiSoft. Among these are **Devpac80** (macro assembler, front-panel debugger, full-screen editor), **Pascal80** (standard Pascal compiler, very fast), **HiSoft C** (Kernighan & Ritchie integer C compiler), **FTL Modula-2** (superb, full Modula-2 compiler with interactive editor), **ZBASIC** (an easy-to-use BASIC compiler), **Nevada COBOL** and **Nevada FORTRAN** (both ideal for learning the languages) and **Write-Hand-Man** (like Borland's Sidekick).

Full details of all these programs and the latest developments are available from:

The Old School
Greenfield, Bedford
MK45 5DE
Tel: (0525)718181