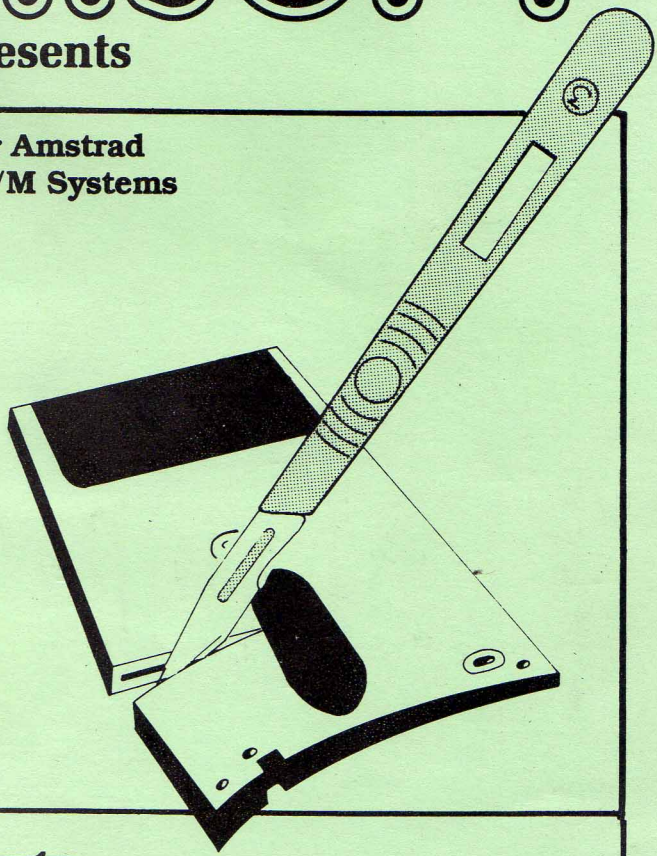
HISOFT

Presents

For Amstrad
CP/M Systems



The Knife

by Aries

THE KNIFE

Welcome!

The Knife is a full-feature disc sector editor and file patcher, allowing you to alter individual bytes anywhere on a disc. Unlike other disc editors about though, The Knife comes as two different programs for two different needs.

If you want to patch an individual file, or if you want to learn about the CP/M disc structure and how files are stored, we recommend you use the first program, called simply **Knife**.

For those of you who want to perform more radical alterations, or want to do something quickly, the more powerful but not quite so friendly **Knife2** program is likely to be more useful.

Although both programs perform approximately the same function, the two are different enough in approach to be explained in separate sections of the manual. The first section is devoted to The Knife, and also explains CP/M disc and directory structures and looks at the parts of this structure specific to the Amstrad implementation. There is also a short example to clarify the concepts, which shows you how to 'un-erase' a file which you've just deleted.

The final part of this manual explains the second program, called Knife2. The tutorial aspects are not covered in such detail in this section, as it is presumed that the user of Knife2 is more familiar with discs. If not, the information contained in part 1 is perfectly applicable to Knife2 as well.

Running The Programs

Both The Knife and Knife2 are supplied on the same disc, but Digital Research / Amstrad licensing agreements do not allow us to provide the CP/M system as well. This means that it is not possible to 'boot' or 'warm boot' CP/M from the disc. Use of the SYSGEN utility, which is provided with the Amstrad machine (or DDI-1), will put a copy of the system tracks onto the disc so that it is possible to boot from it. Please consult the Amstrad or Digital Research documentation for details of how to do this.

It is important to note that neither The Knife or Knife2 may be run from the CP/M Plus operating system. This is because the two operating systems handle discs in slightly different ways, and because Amstrad 'firmware' calls may not be made from CP/M Plus. If you have a CPC6128 or a PCW8256, please boot up CP/M 2.2 before using either version of the Knife.

Getting Started

To start the Knife first make sure that CP/M 2.2 is running then insert the master disc into the floppy disc drive and type after the prompt:

A>knife [ENTER]

The Knife will load in and display its title screen. Press any key (eg [ENTER]) or just wait a few seconds to move on to the first screen. There is a list of options at the top of the screen (often called a menu):

VIEWDISC FILE: EXIT

The VIEWDISC option is highlighted in reverse video. To move to the next option press [TAB]. To select the current option just press [ENTER].

Viewing a Disc

Select the VIEWDISC option by pressing [ENTER]. The Knife then suggests that the disc will be in drive A. Just press [ENTER] to accept this or else type B [ENTER] if you have two drives and want to examine a disc in the second drive.

Now there are some more lines of menu options:

```
 DRIVE:A TRACK:000 SECTOR:00
 FORWARD BACKWARD PRINT CHANGE REWRITE
SEARCH
```

and down below are the contents of the first CP/M record.

The FORWARD option is highlighted and you can step through each record on the disc by repeatedly pressing [ENTER]. You can go back again by using the BACKWARD option (use [TAB] to step round the options. To examine a particular area of the disc, select the TRACK option and type in the track and CP/M sector (also known as a record) number that you require. (Tracks go from 0 to 39 and sectors from 0 to 35).

The PRINT option lets you make a printed copy of a record.

To alter the contents of a sector, select the CHANGE option, make the changes, and then select the REWRITE option. Nothing is written to the disc until you select REWRITE, so you can change your mind at any time before this. After you select CHANGE you can type in the new data that you want in hex using [TAB] to move to the next byte. Move down a line with [CURSOR-DOWN] and you can type in character strings. When you finish making changes, press [ENTER] to get back to the menu.

SEARCH lets you find particular patterns on the disc - such as a text string. Select it and you can then enter the pattern that you are looking for. Before doing so you can enter a 'search mask', which allows some of the bits in each byte to be ignored when looking for a match (eg a search mask of #DF will match regardless of whether letters are in upper case or lower case). Type in the pattern in hex using [TAB] to enter each byte and a dot '.' after the last byte. Alternatively, you can press [CURSOR-DOWN] and then type the pattern as a string of characters - press [CURSOR-DOWN] again to enter some bytes in hex and at the end before typing the dot '.'. The Knife will then start searching for your pattern on the disc, going from sector to sector. You can interrupt the search by pressing any key.

You can return to the top-level menu by using the EXIT option.

Looking at a File

This works in a very similar way to VIEWDISC. Select the FILE option from the top-level menu. An opportunity to select a different user area is given first -this can be defaulted to the normal user 0 by pressing [ENTER]. The records in the file are presented in sequence, and you can use the options described above to examine and alter the contents of the file.

If you intend to perform extensive editing of files, we recommend our Devpac80 package which includes ED80 and MON80. ED80 provides comprehensive Wordstar-compatible text editing. MON80 gives extensive capabilities for editing binary program or data files.

The CP/M Disc Structure

Now that you have your copy of **The Knife**, you can begin to hack away at discs to your heart's content. However, for the program to be really useful, a full understanding of the way CP/M organises files on the disc is most important. Every disc formatted on a CP/M system has a similar structure, and the Amstrad computers' disc system is no exception. Once certain basic disc 'parameters' have been discovered, the organisation of the disc can be relied upon to be thoroughly predicatable.

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

Sectors and tracks

A floppy disc 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. Disc drives get over this problem by moving the head rather than the magnetic film. The disc itself is spinning at a very high speed, and the head is able to move along a radius of the disc, covering any given annulus at any given time. Obviously, a motor is required to move the head across the disc's surface.

We can control motors 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 disc 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 disc. The number of tracks per disc 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 disc, such as used in the Amstrad, this means that the entire working area of the disc 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 disc parameters we mentioned earlier. Likewise, the amount of data which can be held in a sector is determined by another parameter. This is known as the 'sector size'; common sector sizes are 128, 256, 512 or 1024 bytes - the Amstrad discs have 512 byte sectors.

The CP/M Disc Structure

The Amstrad's discs are formatted to have 9 sectors of 512 bytes each per track, and as there are forty tracks, it follows that a side of a disc may hold up to 40 * (9 * 512) bytes which works out as 184320 bytes, or 180K.

When a disc 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 65 (#41) for most of its discs. There are a couple of other formats which use different start sector numbers, but we'll leave the examination of those until later.

With our newly-gained information on tracks and sectors, we can see that it is possible to refer to a given sector's data by its track number and by its sector number. Track numbers always start at 0, so the first sector on a normal Amstrad disc is

Track 0 Sector 65

and the last sector on such a disc is

Track 39 Sector 73

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

(BIGGER POINT SIZE)
The CP/M system believes that sectors are always 128 bytes long. These mythical sectors are also known as 'records', or 'logical sectors', or 'CP/M sectors'. Throughout the rest of this guide we call them 'records'. When the Knife program displays a sector number it is talking about a 'CP/M sector' or 'record'. The real 'physical sectors' on the disc are 512 bytes and we refer to them here as 'sectors'. The Knife program makes no reference to 'physical sectors'. The Knife2 program, however, does exactly the opposite and refers always to physical 512-byte sectors rather than CP/M's logical sectors. It does this by 'grouping' four consecutive logical sectors.
(OK. BACK TO NORMAL POINT SIZE)

CP/M Discs

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 discs. This means that every CP/M file on a CP/M disc is an exact multiple of 128 bytes in length. Now read again the bold 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. On the Amstrad disc system, there are eight records (or 1024 bytes) per block, which

means that a block occupies two physical sectors. The sectors will be contiguous in numbering terms, and as the first block occupies sectors 65 and 66 of track 2 the second block occupies sectors 67 and 68 of the same track.

Files always occupy a whole number of blocks, even if the physical length of a file would seem to dicate otherwise. This means that a file which is five records long will occupy exactly the same amount of disc space as one which is seven records long. It is clear that the smaller the block size, the more efficient the use of disc 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 discs are now rather cheap, a large block size is often adopted.

The directory of a CP/M disc 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 disc 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 disc parameters are required. The information we need is held on page 2.8 of the Amsoft DDI-1 firmware manual (SOFT 158A). This is what it tells us:

System format (this is the one we're interested in)

Single sided
512 byte physical sector size
40 tracks numbered 0 to 39
1024 byte CP/M block size
64 directory entries
9 sectors per track numbered #41 to #49 (this is 65 to 73 in decimal)
2 reserved tracks

Some of this information is not new to us, but the part that gets us to the directory is the '2 reserved tracks' entry. A CP/M system disc always has some reserved tracks, to hold the code required to boot the system into the machine. These tracks also generally hold the CCP (Console Command Processor) and the BDOS (Basic Disc Operating System). Reserved tracks always start from the beginning of the disc, 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 must be

Track 2 Record 0

If you examine this record, you'll see that we're right! What do you notice about the organisation of this area? Let's examine the first few bytes of a typical directory record to find 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 disc structure. You'll remember that the system's disc parameters mentioned that there are 64 directory entries. If each directory entry is 32 bytes long, the entire amount of disc 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 disc occupies records 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 disc, 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 (229), the file has been deleted. A deleted file's directory entry can be overwritten by the next file to be created on the disc, and all files whose first directory byte is #E5 will not be found in an open call.

As well as 0 or #E5, the first byte can 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 give the built-in USER command to move to another user area. The main use of this is on large (hard) discs 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 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 will not be able to access the file!

The '.' dot which we use to separate filenames and extensions, as in

```
GEN80 MYFILE.GEN
```

are 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 'extents' 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 eight records per block, it follows that PIP.COM will occupy eight 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 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. From this, we can see that PIP.COM occupies blocks 2, 3, 4, 5, 6, 7, 8 and 9. We also know that all eight records in blocks 2 to 8 are part of PIP, but only the first two 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 disc, and thus locate every record of a file. There is one problem, though...

Extents

As the last sixteen 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're 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 (on the Amstrad).

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 am Amstrad CP/M or Amsdos disc, 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.

The final pieces of information held in a directory entry are cleverly hidden away in the 3-byte filename extension field. These are normally capital letters or other ASCII characters, which means they are below #80 (128), which in turn means that the very top bit, bit 7, is zero. 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 directory listings.

Other Amstrad Disc Formats

Although the principles we have discussed apply to all discs formatted under a CP/M system, it must be borne in mind that different systems will have different disc parameters; there may be thirty two records to a block, or a sector size of 256 bytes, maybe even only eight sectors per track. Some CP/M systems may seem at first glance to differ wildly to our description, but all they are doing is expanding upon the principles. Nevertheless, by getting hold of this basic information, the rest is made simple.

Amstrad itself allows for two other disc formats apart from the 'system format' discussed above. There is the 'Data Only format', which is identical in every respect to the system format except that there are no reserved tracks, and the sectors in a track are numbered from 193 to 201. The other format, which is used very infrequently, is the 'IBM format'. This is designed to be compatible with CP/M-86 systems used on the IBM PC, and has only eight sectors per track, one reserved track and sectors numbered from 1 to 8.

An Example - How to Unerase a File

Now that we know how to use the Knife, and also what the

information on a disc means, we are ready to try an example. We are going to recover a deleted file: perhaps by over-zealous use of the ERA command. Note that you cannot recover files after other data has been written on the disc - you must do it straightaway.

The first step is to find the directory entry or entries for the file, so select VIEWDISC, and goto TRACK 2 SECTOR 0. Now SEARCH for the name of your file, using a MASK of hex DF. Suppose the name is "count.gen" so we set up the search string by typing:

[CURSOR-DOWN] C O U N T [SPACE] [SPACE] [SPACE] G E N [CURSOR-DOWN]
(don't forget the dot at the end)

The Knife now searches for the directory entry and hopefully finds something like the one we showed you earlier. We can see that the first byte of the directory entry (just before the filename) is E5. All we need to do is change this to be a zero and our file will be magically restored. Select the CHANGE option, use [CURSOR-DOWN] to move to the appropriate byte, type 00 to change it, then [ENTER] to return to the menu, and select REWRITE. The file is restored! For large files you must remember to change the directory entry for each extent in the file.

Knife2 is an alternative disc editor to the Knife, offering different facilities and approaching the idea in a completely different way. We think that the two programs complement each other, which is why we supply them both.

One of the most important differences between the two Knives is that this version (referred to always as **Knife2**) refers always to **physical** sectors (512 bytes on Amstrad discs) rather than CP/M's **logical** sectors.

Knife2 is loaded by typing its name at the CP/M prompt:

```
A>knife2
```

Due to the major differences in BIOS (Basic Input/Output System) operation between CP/M 2.2 and CP/M Plus, Knife2 is capable of running successfully only in a CP/M 2.2 environment. For CPC464 and CPC664 owners, this makes no difference, but owners of later machines such as the CPC6128 will need to boot up CP/M 2.2 rather than CP/M Plus. Don't worry though - Knife2 can still be used to edit CP/M Plus discs. If you try to execute Knife2 while running CP/M Plus, the beeper is sounded and this message appears:

```
This program must be run under CP/M 2.2
```

You are then returned to CP/M.

Once the program has loaded, the screen is cleared and you will be asked which drive the disc to be edited is in. This may be either 'A' or 'B' on current Amstrad systems. The program may also be terminated at this point by pressing [RETURN] rather than 'A' or 'B'. If the drive specified does not contain a disc, does not exist or contains an unrecognised disc format (very unlikely!), a message is printed and you are asked to specify the drive once again.

When a disc has successfully been selected, the first 256 bytes of the first sector on the first track of the disc are displayed, something like this:

```
DISC SECTOR EDITOR V1.1 (C) 1985 HiSoft
    Drive: A     Track: 00     Sector: 01

0000  HH 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 HH  AAAAAAAAAAAAAAAA
0020  HH 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 HH  AAAAAAAAAAAAAAAA
0040  HH 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 HH  AAAAAAAAAAAAAAAA
0060  HH 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 HH  AAAAAAAAAAAAAAAA
0080  HH 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 HH  AAAAAAAAAAAAAAAA
00A0  HH 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 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
```

There are sixteen bytes to each line, and the first four digits on each line represent the hexadecimal offset from the start of the sector of the first byte in each line. Following this offset are the hexadecimal representations of the sixteen bytes and then the ASCII (character) representations of each of the bytes. Note that the ASCII representations are coerced into 'printable' characters by first bitwise-ANDing each byte with 7F (127 decimal) and then checking to see if the result forms a **control code**. If it does then it is replaced in these columns with a full stop. A control character is one with a code value of less than 20 (32 decimal) or equal to 7F (127 decimal).

The reason that it says 'First Half' at the bottom is that each sector on an Amstrad disc is 512 bytes long. As it is difficult to get all these bytes on the screen in one go, we show only a half at a time. Switching to the other half is simple, as we shall see.

When this display has been drawn, a cursor will appear over the first digit of the first byte. At this stage you may enter commands to be acted upon by Knife2, or alter any of the bytes displayed by typing in hexadecimal numbers.

Altering Bytes

When the cursor is over the hexadecimal representations of the bytes, new values may be entered simply by typing them in. Any valid hexadecimal digit will be entered into the current location and the cursor advanced one position. Valid hexadecimal digits are '0' to '9', 'A' to 'F' and 'a' to 'f'.

When the cursor is over the ASCII representations of the bytes, new values may be entered by typing the required character. As a few of the keys need to be used to issue commands to the program, not all ASCII values may be typed in directly. In these cases, move the cursor to the hexadecimal representation of the byte and enter the hexadecimal value of the character.

In either case, the cursor may be moved around the hexadecimal or ASCII representations with the four cursor keys (those with arrows on). You may switch between the hexadecimal and ASCII representations at any time by pressing the **TAB** key or **CONTROL-I**.

Remember that no alterations you make are saved back to disc until you issue a 'Write sector' command (see below), so it is perfectly permissable for you to play around with the system until you get used to it.

Commands

Knife2 responds to a wide variety of commands, many of which are accessed by pressing the [CONTROL] key in conjunction with another. This means that the two keys must be pressed simultaneously, but it doesn't matter if the other key is upper- or lower-case.

SHIFT-Right arrow — **GOTO next sector**

Pressing this key combination advances to the next sector on the disc, updating the display as appropriate. If the current sector is the last on a track, then the first sector of the next track is read. If the current sector is the last of the last track, this command goes to the first sector on the first track.

SHIFT-Left arrow — **GOTO last sector**

Pressing this key combination retreats to the previous sector on the disc, updating the display as appropriate. If the current sector is the first on a track, then the last sector of the previous track is read. If the current sector is the first of the first track, this command goes to the last sector on the last track.

SHIFT-Down arrow — **GOTO next track**

Pressing this key combination advances to the next track on the disc, updating the display as appropriate. The current sector number is not changed. If the current track is the last on the disc, this command reads the first track on the disc.

SHIFT-Up arrow — **GOTO last track**

Pressing this key combination retreats to the previous track on the disc, updating the display as appropriate. The current sector number is not changed. If the current track is the first on the disc, this command reads the last track on the disc.

COPY — **Show other half**

Pressing [COPY] re-draws the display using the other half of the sector as its data. If the first half of the sector is being displayed, then this key causes the second half to be displayed, and if the second half is currently being displayed, this key will cause the first half to be shown. Nothing else is affected.

TAB — **HEX/ASCII switch**

This key moves the cursor between the hexadecimal representations and the ASCII representations of the bytes being displayed. The cursor's effective position within the data is not altered.

ESC — **Quit edit and leave**

Pressing [ESC] causes the current editing session to terminate, returning you to the prompt asking which drive the disc to be edited is in.

CONTROL-F **Search for a string**

This key produces a prompt at the bottom of the screen. Up to eighty characters may be typed at this prompt, and the normal CP/M editing keys may be used while it is being entered. When eighty characters have been entered, or when you press [RETURN] (whichever is the sooner), the program begins to search for the string on the disc starting from the sector following the one being displayed. If the string is found, the sector and track in which it occurs become the current ones, and the display is updated to reflect this. If the string cannot be found, a message is displayed. Pressing a key at this point returns the display to what it was prior to [CONTROL-F] being pressed. The search may be aborted at any time by pressing any key. If this is done, the sector which was being searched at the time the key was pressed is made the current sector.

Notice that the string to be searched for must exist wholly within a physical 512-byte sector for it to be found.

CONTROL-G **Search for a byte sequence**

This key produces a prompt at the bottom of the screen. Up to eighty bytes (values between 0 and 255) may be typed at this prompt, and the normal CP/M editing keys may be used while each is being entered. Each byte must be separated by a press of the [RETURN] key. The program expects each byte to be entered in hexadecimal, but this may be over-ridden by preceding it with a '#' character. When eighty byte values have been entered, or when you press [RETURN] by itself (whichever is the sooner), the program begins to search for the string of bytes on the disc starting from the sector following the one being displayed. If the string is found, the sector and track in which it occurs become the current ones, and the display is updated to reflect this. If the string cannot be found, a message is displayed. Pressing a key at this point returns the display to what it was prior to [CONTROL-F] being pressed. The search may be aborted at any time by pressing any key. If this is done, the sector which was being searched at the time the key was pressed is made the current sector.

Notice that the sequence of bytes to be searched for must exist wholly within a physical 512-byte sector for it to be found.

CONTROL-Q **Write sector to given location**

This key allows you to write the sector being edited back to any location on the disc. Unlike [CONTROL-W], which writes the sector back always to its original location, this keypress asks you for new track and sector numbers and then writes the 512-byte sector to the disc at that location. If [RETURN] alone is pressed in

response to either 'New sector number:' or 'New track number:'; the current value for that parameter is used. Inexperienced users must be very careful with this command, as it is obviously easy to entirely foul up the disc if the sector is written to an undesired location.

CONTROL-S **Select new sector**

Pressing this key results in you being asked for a new sector number to edit. If [RETURN] alone is pressed in response to this prompt, the sector number is not altered. The sector number is normally between 1 and 9. This command allows you only to select sectors within the current track.

CONTROL-T **Select new track**

Pressing this key results in you being asked for a new track number to edit. If [RETURN] alone is pressed in response to this prompt, the track number is not altered. The track number is normally between 0 and 27 (0 and 39 in decimal). This command does not alter the number of the number of the current sector within the track.

CONTROL-W **Write sector back**

This command writes the 512 bytes being edited back to disc, at the location they originated from.

Sectors And Tracks

CP/M 2.2 regards each disc as comprising of a number of 'logical sectors', each of which is 128 bytes long. The Amstrad implementation of CP/M has 36 of these logical sectors on each track, and as each 'physical sector' on these discs is 512 bytes long, it follows that there are four logical sectors to each physical sector. From this, we can see that each disc must have nine sectors on each track. When discs are formatted, the physical sectors may be given odd numbers; for example, the Amstrad CP/M 'System format' discs have sectors numbered from $41 to $49. To avoid all the confusion, we map the CP/M logical sectors to each physical sector, but always treat the first sector in each track as being sector one. It makes no difference in practice, but it may help to remember that Amstrad sector number $41 is Knife2 sector $01.

Track numbers are always between 0 and 27 (0 and 39 in decimal), as all Amstrad discs are currently 40 tracks wide.

Knife2 is aware always how many sectors per track there are on a disc, and although this is almost always nine, the Amstrad 'IBM format discs' have only eight.

Other Products

While The Knife and Knife2 allow you to alter individual bytes on discs and examine the structure used by the operating system, text files may be far more efficiently created and altered with a full-screen editor. As part of the package provided with each of our CP/M language products, HiSoft's screen editor **ED80** is the ideal to generate language source files, letters and program documentation.

Our range of languages is well respected and extremely popular. We have a full function Pascal compiler which follows the Jensen-Wirth standard very closely and is especially tailored to writing effective CP/M programs. Our new CP/M C compiler is one of the cheapest available, yet offers all the facilities of the language expected. These include structures, Unix-style stream I/O, byte-addressable random-access files and full overlay support. This compiler, which includes an extensive function library and the ED80 screen editor, is destined to become the most popular CP/M C compiler. A similar variant is available as an AMSDOS-only compiler.

For those who want to access those aspects of CP/M and AMSDOS available only through machine code, Devpac80 comprises a full Z80 macro assembler, a front-panel debugger with full breakpoint and single-step capability, and the full screen editor.

HISOFT
180 High Street North,
Dunstable, Beds LU6 1AT
(0582) 696421

HISOFT
180 High Street North,
Dunstable, Beds LU6 1AT
(0582) 696421

Printed by Jiffy Print Limited, Luton, Beds.