

WITH A **COMMODORE 1987**

DISK USERS HANDBOOK

**BEGINNERS GUIDE
TO DISKS**

**PROGRAMMING WITH YOUR
DISK DRIVE**

**USING A
DISK
EDITOR**

**FAST LOADER
FOR 1541**

**C64 DISK
EDITOR**

**PRINT YOUR
OWN
DISK SLEEVES**



Free with the December 1987
Issue of Your Commodore

Your Commodore Proudly Presents



Available from newspapers and by mail-order from



Informat Ltd, 5 River Park Estate, Berkhamsted, Herts HP4 1HL. Tel: 04627 7666/1/4 (£1.50 plus 50p p&p)

Editor: Stuart Cooke
Assistant Editor: Sue Joyce
Editorial Assistant:
Kirk Rattler
Advertisement
Manager: Stuart Taylor
Advertisement Copy
Control: Laura
Champion
Origination: Ebony
Typesetting
Design: Argus Design
Studio

Your Commodore
Incorporating Your 64 in a monthly magazine appearing on the first Friday of each month. Your *Argus* is published every second month within the pages of *Your Commodore*. Argus Specialist Publications Limited, Editorial & Advertisement Office, Your Commodore, No 1 Golden Square, London W1R 1AB. Telephone: 01-477 9020 Telex: 864 0000.

Subscription rates upon application to Your Commodore Subscriptions Department, Infocast Ltd, 5 River Park Road, Northwood, Herts, SG4 1HL.

**ARGUS
PRESS
GROUP**

The contents of this publication including all articles, designs, drawings and programmes are copyright and other intellectual property rights therein belong to Argus Specialist Publications Limited. All rights reserved by the Law of Copyright and other intellectual property rights and by virtue of international copyright conventions are specially reserved to Argus Specialist Publications Limited and any reproduction requires the prior written consent of the Company. © 1987 Distribution: BHI 4 1187 Distribution: BHI 4 1187 Distribution: 16-14 Trinity Gardens, London SW9 0JL. Printed by Chase Web, Plymouth. Opinions expressed in reviews are the opinions of the reviewers and not necessarily those of the magazine. While every effort is made to thoroughly check programmes published for errors we cannot be held responsible for any errors that do occur.

CONTENTS

Beginners Start Here	4
Discover the benefits a disk drive can bring to your system.	
Disk Commands	7
Understanding Direct Access Commands	
Disk Editing	12
You can recover scratched files and much more when you can edit disks.	
Track Sector Editor	19
How to make full use of your editor.	
DIR Cover	24
Stop wasting time to find a specific program - create a handy check list	
1541 Fast Loader	27
Speed up the notorious 1541 disk drive.	
Menu Maker	31
Don't rely on memory - make a menu	

The *Your Commodore Disk Users Handbook* is packed full of vital information and programs for owners, and potential buyers, of all Commodore disk drives.

If you are new to your disk drive then our beginners article will supplement the manual and help you discover the joys of using a disk drive. If you are a more advanced user, the article on disk commands will expand your knowledge so that you can talk directly to the drive. Learn how to read a directory from within Basic and much more.

For those readers wanting to go even further with their disk drive we give a detailed description of the disk structure and details of how to use this information with a disk editor, including how to resurrect scratched files.

The programs

As well as the articles already mentioned, this supplement also holds a variety of useful programs. Owners of the Commodore 1541 disk drive, which has often been described as a lumbering hippo, can speed loading up with our 1541 Fast Loader.

Should your appetite be whetted by our Disk Editing article then you can type in our Track/Sector Editor and give it a whirl yourself.

Should you have a large number of programs in your disk collection then our fast run programs will be invaluable. The first, DIR cover, will produce your own disk library stories on your printer. The program is totally in Basic and can therefore easily be altered to suit any printer.

The second program is a C64 menu generator. This will place a menu of selected files on your disk. You can then use this menu to load any of the selected programs with ease.

All in all, whatever your technical ability, the *Your Commodore Disk Users Handbook* will provide you with something to suit your needs.

Beginners Start Here

If you're the proud owner of a disk drive, you'll already be discovering the benefits it can bring to your system.

Read on for more info.

By Tony Hetherington

Congratulations, if you've just bought a disk drive, then you're at last free of the frustration of waiting 15-20 minutes for a game to load. At last you can load and save your own programmes in seconds and can access the huge library of disk-based software. If you haven't already added a disk drive to your system but are wondering whether it would be worthwhile, then read on as we delve into the delights that lie ahead.

Apart from the considerable reductions in loading time, which is worthwhile on its own, a disk based system means you can now use bigger programs as although they can't use the whole of the disk storage space (about 170K) at the same time information can be loaded in as and when required. If you don't think 170K is enough you could run up to four disk drives from your C64 at any one time or store your data on more than one disk.

Finally, a disk based system is a lot more flexible than a cassette because as there is two-way communication between your C64 and the disk drive, any piece of information on a disk can be quickly read, altered and rewritten in a few seconds. This is why nearly all business software such as word

processors, databases and spreadsheets are disk based.

What is a disk?

A disk is a flat disc of magnetic material made from a thicker version of the material used to make cassette tapes. The disk is then sandwiched between two sheets of special material that gently cleans any dirt off, as it spins in the disk drive. This is then sealed in a plastic cover to protect it from scratching, and any dirt, grime and grease that could damage the disk by handling it.

The plastic sleeve has several custom notches which allows the disk drives head to read the information on the disk, slots to guide the disk to the correct place in the drive and a write-protect notch or hole. The drive senses the hole and allows new information to be written to the disk. Since this can mean writing over important data you can 'write protect' a disk by sticking a tab over the notch. This tells the drive to stop any commands that would write on the disk. Most blank disks are supplied with a sheet of write protect labels.

It's worth taking care of your disks

as damaging part of one could ruin the whole disk. After all, 170k of data is a lot to lose! The following tips are worth following as they could save you a lot of time and spare you a lot of inconvenience.

- 1 Only touch the plastic sleeves and handle disks gently at all times. A bent disk is a ruined disk.
- 2 When you're not using a disk keep it stored in its cardboard (or heavy paper) sleeve and preferably in a plastic disk box.
- 3 Keep disks away from bright sunlight, cigarette smoke, coffee, dust, telephones, monitors and the top of the disk drive or other sources of magnetic fields.
- 4 Don't take a disk out of the drive when the red light is on as this means the drive is reading or writing information and could cause you horrendous problems.
- 5 Always ensure that you take disks out of your drive before you switch it off as you run the risk of losing everything on it.

Disks are supplied in a variety of sizes and are labelled to show the amount of information that can be stored on them. All disks are

manufactured to be "double sided, double density" and are then tested for quality. If they fail these stringent quality control tests, they are then down graded to single-sided, double-density or double-sided single-density. The Commodore 1540 disk drive only requires single sided, single density (SSSD) disks which means you don't have to waste money on extra quality you won't need.

As mentioned before all disks are originally manufactured to the double-sided and so you can buy a small device known as a disk wiper (for around £3) that will cut a second write protect notch into the disk so you can then use the other side! Obviously, there are no guarantees that this extra side will always read and write data perfectly but knowing it can be used is useful to know.

When you buy a disk it is a completely blank disk of magnetic material and so must be prepared for use with your C64. This is required since the same disk could have been bought by someone to use in a IBM or an Atari computer. Therefore, the first thing you must do is prepare or format it for use.

To format a blank disk ... place it in the disk drive and then do these three steps in the following command:

```
OPEN 8,15,"N:\diskname.D"
```

This command tells the processor inside the disk drive to open communication channel 1 (this can be any of 15) to device number eight (the disk drive). The 15 tells the drive that the rest of the command is an instruction for the whole disk and tells it to format the disk and give it the name diskname which is followed by the disk ID. The ID is a two letter or number identity code to distinguish the disk from other disks with the same name. For example, you could name a whole series of disks Tony.H1, Tony.H2, Tony.H3 and so on. Therefore the command to name a disk Tony.H1 would be ...

```
OPEN 8,15,"N:\Tony.H1"
```

This should then be followed by CLOSE 1 to close the number 1 command channel.

When this command is entered, the disk drive will whir into action and the red light will flash on and off. This will take a few minutes as the drive has a lot to do. First of all, it creates 35 circular tracks on the disk and divides these into sectors or blocks. Because

the circumference of a disk is wider at the outside than the inside there are more blocks on the outer than the inner tracks.

Each block can contain 256 characters of information although the first two characters are used by the drive to point to the next block where information is stored. Once each block has been created the drive tests it and then finally adds a directory in the center of the disk which contains a list of all files or programmes stored on the disk and a Block Availability Map which helps the drive offer new information into empty blocks.

When this process is completed the drive will stop and the disk will be ready to use.

As you might imagine, formatting a disk wipes all information that was stored on the disk so you should be careful that you don't format any disks that contain information you still need and NEVER format a disk containing a commercial program.

Device Numbers

The format and Load and Save commands include the device number 8. This tells the C64 which input or output device the information should be read or written to. The C64 uses the following device numbers:

- 1 - cassette
- 2 - keyboard (input only)
- 3 - Screen
- 4a? - Printer (usually 4)
- 8a? - disk drives.

Most people will only use one disk drive which is automatically set to device number 8. However, if you have a second drive (or third and fourth) and want to use it at the same time you will have to give it another device number (usually 9).

You can do this in two ways either by altering the hardware or a simpler way is to type in and run the program in the manual.

Loading and Saving

Now you have prepared or formatted a disk for use or have bought a commercial program you will want to load and save programs.

To load a program simply type ...

```
LOAD"filename".S
```

which loads the name program into

memory. Then type RUN to start it. OR type

```
Load"filename".S
```

which loads the first program on the disk into memory.

OR

```
LOAD"filename".S,1
```

loads the first program on the disk into the same memory location it was saved from. This is the command you will use most for commercial programs which usually start automatically.

OR

```
LOAD"filename".S,1
```

This means that the first program on the disk is loaded in. Occasionally Load"filename".S,1, if used a few times will load in the next program on disk.

The command Load "S",S loads in the disk directory that can be displayed by typing LIST which shows all the files that are stored on the disk.

To SAVE a program simply type

```
SAVE"filename".S
```

You will only need to use this if you're going to write and use your own programs as commercial programs have their own save routines but you will still need to ensure that you have a formatted disk ready for use.

The asterisk (*) which can be used in loading commands replaces any number of characters. The command LOAD"filename".S loads in the first program on the disk as the * replaces the filename. You can also use * to save your typing finger and load in files further down the directory. For example, if there was a program called HOW TO USE THIS you could load it in by simply typing the command LOAD"HOW".S as long as there wasn't another program listed above this one in the directory called HOW 1 WDN.

File Types

As mentioned above, typing LOAD"filename".S then LIST displays the disk directory on screen. As you can see from these examples there are four different types of disk file.

The program file which appears as PRG in the directory listing is probably the most common file that you will come across. A program file is exactly what its name suggest, a program that you have scored on disk.

The program file is stored on disk in exactly the same format as it would be in the computer's memory, i.e. it is taken as is.

A sequential file (SEQ) in the directory listing) is essentially a file that contains a continuous string of characters. A sequential file could, for example, be set up to contain data for a database. Let's say that we had two names in our database - Fred Higgs and John Smith. In a sequential file the data would be stored as:

John Smith Fred Higgs.

In other words as a continuous list. The problem with using this type of file to store data is that if you required, say the 50th entry of a database, the previous 49 entries would all have to be loaded in. This makes access to your data very slow.

A much better type of file to use for data storage is a relative (REL) file. This type of file allows you to select a specific record, delete a specific record. In other words you can access the information that you require from the file without having to read lots of unwanted data into memory. Perhaps the best used type of file is the user (USR) file. This is really just like a sequential file and is used in the same way.

Housekeeping

As you save and load files to a disk it will rapidly fill up with things you no longer need. The following commands allow you to tidy up your disks and to save disk space.

New

The new command will look familiar as one of its forms is the same as the format command.

OPEN1,8,15,"N:diskname,JD"
(The N is short for NEW). This wipes the disk and marks out the sectors and tracks.

If the disk has been used before you can shorten this by leaving out the JD. This may not sound a lot but the process is shorter as the drive doesn't have to recreate each block.

Initiate

If you are writing your own programmes and want to use a second disk then you must use the initiate

command to tell the drive that you've swapped disks and instruct it to read in the new RAM.

Typing **OPEN1,8,15,"1"** initialises the new disk ready for use.

Scratch

If you find you've a program on disk that you no longer need, such as an earlier version of an existing program then typing

OPEN 1,8,15,"0:filename" will delete it.

You could use the * to delete everything, but when you're deleting files I find it best to type out the full name as you're less likely to make a mistake and delete a file you desperately needed.

Validate

Once you've saved and deleted a few files the blocks of each file will be spread about the disk. This won't stop the files from being read or written but it will slow down the process as the drive head must move over each block.

Typing **OPEN1,8,15,"V"** will start the spring cleaning process. This can take some time but will be worth it as you'll be surprised at the saving in loading times.

Disk Commands Summary

The following commands are entered through the command channel. For example **OPEN1,8,15** followed by...

NEW/Format - "N:diskname,JD"

Scratch - "0:filename"

Initiate - "1"

Validate - "V"

Disk Software

Now you have a disk drive you can use a variety of disk-based business packages, games and utilities.

A disk drive is essential if you plan to use one of the many business packages on the market. Although there are one or two tape based word processors, their disk based counterparts offer far more facilities such as different typefaces or fonts, the ability to include graphics in your text, quick loading and saving and even a spelling checker to correct any mistakes.

As well as create, merge, edit, load and save text files you can get your figures and budgets right with a spreadsheet or store thousands or records with a database program such as Superbase 64.

GEOS (Berkley Softworks now available through Microsoft) brings icons and pull down menus to your C64 as well as a whole new disk operating system. With GEOS you don't have to type in long commands you simply point at an icon instead. There is also a special GEOS word processor, a spreadsheet and database programmes.

If you want to be entertained then why not try a disk based game. There are often extended versions of the classic games but have added features or more rooms, courses and options. As well as improving existing games you can also delve into the disk only world of the Infocom adventures. These are amazing games packed with mind boggling puzzles and text descriptions to fire your imagination. Whatever your particular interests you'll be able to find at least one Infocom adventure to explore. Fantasy fans can explore the amazing Zork trilogy and the worlds of Enchanter, Sorcerer and Spellcaster or Sci-Fi buffs can step boldly into Planetfall and Starfall or even enter the amazing Hitch Hikers Guide to the Galaxy!

SSI games are rarely converted onto tape and so you need a drive to command great battles in one of their many wargames or explore the roleplaying games such as Phantasy and Shards of Spring. Or why not try Rainbow's graphic adventures such as The Pawn and The Guild of Thieves or the amazing Ultima series of games. The next one to reach these shores will be Ultima V and will be so big that it will cover both sides of four disks! Try and get that on tape!

Just when you're thinking how much faster your disk drive is than the cassette player someone, somewhere develops a program that makes it even faster. Cartridges such as Quickdisk and the Expert will speed up your drive and also include a fast disk formatter but the most impressive device is Dolphin DOS from Evidash-Micros. This is actually an entirely new disk operating system that replaces the existing DOS in your drive and will allow you to load programs in seconds and not minutes!

Disk Commands

Learn how to use your disk drive more efficiently.

By Stuart Cooke

Not only do Commodore disk drives provide the user with commands to format the disk, read the contents of the disk, LOAD programs on, there is also a whole range of less documented commands that allow you to talk to the disk and disk drive directly. This range of commands is referred to as Direct Access commands. Once you understand the concept of these commands and how the disk drive works you can get the drive to do whatever you want.

Inside the 1541

Probably the most common of the whole range of Commodore drives is the 1541. For the sake of this article we will refer to this drive. Most of the information is the same for all of the other drives.

Before we take a close look at the direct access commands that are available it is about time we had a look at the inside of a 1541 disk drive. Figure 1 is a memory map of the disk drive. Before you can program the disk drive efficiently it is important that you know its inner workings.

Talking to the drive

Now that you've had a close look at what you can get at inside a drive it's

time to move onto the direct access commands.

All communication between the disk and the user is made through a buffer. If you take a look at Figure 1 you will see that there are five buffers available. However, only four of these are free for use. Buffer four is normally reserved for holding an image of the disk RAM. When using SIX and REW files at the same time, buffer number three is also not available because the directory uses it.

If you want to write information onto the disk or read information from it then the sector that you want to manipulate must be read into one of the buffers. When you wish to use a buffer, you first have to OPEN a channel and specify which buffer you wish to use. For example OPEN 1,0,2,"*2" would open the channel to buffer number 2. However, it is good practice not to specify the buffer number but let the DOS select it for you. You achieve this by not specifying a number after the "*" sign. For example:

```
OPEN 1,0,2:"*"
```

If your selected buffer contains alphanumeric data, and is not over 88 characters in length you can use the INPUT # command to read in data from the buffer. Otherwise you will have to use the GET # command. Note that when using GET # it does not check for null characters. It is

therefore advisable to have the following basic line, or something similar, inside a program that reads data from the disk with a GET # statement.

```
IF AS="" THEN AS=CHR$(0)
```

Obviously the character read from the disk must be stored in AS.

Before we go any further there are four things that you should remember.

- 1 A PRINT # command to the command channel (secondary address of 15) sends direct access commands to the DOS.
- 2 A PRINT # statement to any other channel (i.e. secondary address not 15) sends data into one of the buffers already mentioned.
- 3 An INPUT # or GET # statement to the command channel (secondary address of 15) returns any error messages.
- 4 An INPUT # OR GET # statement to any other channel reads data from one of the buffers.

Block-Read

The block-read command talks the 1541 to read a sector from the disk into your open buffer - strictly speaking this is known as a direct access file. This command is shortened to "WRD" when talking to the drive or should you prefer to shorten the command even more, use the command "UR". An

1541 Memory Map

DRIVE ADDRESS		Description
HEX	DEC	
#0000	0	Command code for buffer 0
#0001	1	Command code for buffer 1
#0002	2	Command code for buffer 2
#0003	3	Command code for buffer 3
#0004	4	Command code for buffer 4
#0008-0007	8-7	Track and sector for buffer 0
#0008-0008	8-8	Track and sector for buffer 1
#000A-000B	10-11	Track and sector for buffer 2
#000C-000D	12-13	Track and sector for buffer 3
#000E-000F	14-15	Track and sector for buffer 4
#0012-0013	18-19	ID for drive 0
#0014-0015	20-21	ID for drive 1
#0018-0017	24-23	ID
#0020-0021	32-33	Flag for head transport
#0030-0031	48-49	Buffer pointer for disk controller
#0038	57	Constant 8, mark for beginning of data block header
#003A	58	Parity for data buffer
#003B	61	Drive number for disk controller
#003F	63	Buffer number for disk controller
#0043	67	Number of sectors per track for formatting
#0047	71	Constant 7, mark for beginning of data block header
#0049	73	Stack pointer
#004A	74	Step pointer for head transport
#0051	81	Actual track number for formatting
#0069	108	Step size for sector division (1/3)
#006A	109	Number of read attempts (2)
#006F-0070	111-112	Pointer to address for H and B code.
#0077	119	Device number plus #20 (#22) for listen
#0078	120	Device number plus #40 (#44) for talk
#0079	121	Flag for listen (I/O)
#007A	122	Flag for talk (I/O)
#007C	124	Flag for ATM from serial bus receiving
#007D	125	Flag for EDI from serial bus
#007F	127	Drive number
#0080	128	Track number
#0081	129	Sector number
#0082	130	Channel number
#00A3	131	Secondary address
#00A4	132	Secondary address
#00B8	133	Data byte
#00BB-00BD	139-141	Work storage for division
#00C4-00C5	148-149	Actual buffer pointer
#00C8-00C9	152-153	Address of buffer 0 #0300
#00CB-00CC	155-156	Address of buffer 1 #0400
#00CD-00CE	157-158	Address of buffer 2 #0500
#00CF-00D0	159-160	Address of buffer 3 #0600

#00A1-00A2	101-102	Address of buffer # 00700
#00A3-00A4	103-104	Pointer to input buffer #0200
#00A5-00A6	105-106	Pointer to buffer error message #02D5
#00B5-00BA	101-106	Record number lo, block number lo
#00BB-00CD	107-102	Record number hi, block number hi
#00CL-00CE	103-104	Write pointer for REL file
#00CF-00CC	105-204	Record length for REL file
#00D4	213	Pointer in record for REL file
#00D5	213	Side sector number
#00D6	214	Pointer to data block in side sector
#00D7	215	pointer to record in REL file
#00E7	231	File type
#00F0	240	Buffer number
#0100-0145	250-225	Stack
#0200-0220	313-303	Buffer for command string
#024A	306	File Type
#0250	300	Record length
#0250	301	Track side sector
#025A	302	Sector side sector
#0274	323	Length of input line
#0278	332	Number of filenames
#02D7	340	File control method
#02E0-02E4	340-344	Track of a file
#02E5-02E9	345-349	Sector of a file
#02E8-02F0	705-701	Buffer for error message
#02FA-02FC	702-704	Number of free blocks
#0300-03FF	705-1023	Buffer 0
#0400-04FF	1024-1279	Buffer 1
#0500-05FF	1280-1535	Buffer 2
#0600-06FF	1536-1791	Buffer 3
#0700-07FF	1792-2047	Buffer 4

Fig. 1

example of how to use the command is shown later. As a point to note, some Commodore drives have a bug in the B-R command and for this reason, it is always best to use the "U1" command.

Block-Write

The block-write command is the exact opposite to the block-read command. This takes the contents of the buffer in use and writes it into the specified sector. The format for this command is B-W or U2. Again a problem exists with B-W so use the U2 command.

Allocating Space

The Block-Allocate, or B-A, command allows the user to reserve blocks on the disk. The main purpose for this is to reserve areas of the disk

for special usage. The Block-Allocate commands clear the necessary bits in the Block Availability Map after execution of this instruction.

The Buffer-Pointer command, shortened to B-P, tells the DOS just where you wish to start reading or writing data to or from in a buffer.

When using the direct access commands there are two formats for the command available. Either may be used depending upon your own preference. The first method is:

```
PRINT #15,"U1,"channel-number;
drive
```

the second method is:

```
PRINT #15,"U1 channel-number;
drive"
```

Now that we've discussed what commands are available, let's take a closer look at them in use. The

following examples should make the use of buffers and direct access commands much clearer.

Suppose you wished to follow a program through an disk by track and sector without actually making in any data. To do this you need to follow the path of the "link" bytes. That is, the two bytes at the start of each block that tells you where the next track and sector of the specific program is.

The program in Figure 2 gives an example of how you would perform this task.

For our second example let's presume that we wish to read the diskette name from within a program. As you already know, our article on disk editing, the name starts at position 144 of track 18 sector 0. Using a B-R command you would read the specified sector into the buffer. You

```

1 OPEN#0,8,15 : REM OPEN THE
COMMAND CHANNEL
2 OPEN 4,8,4,"#": REM OPEN
DIRECT ACCESS FILE
3 INPUT "TRACK AND SECTOR
PLEASE";TR,SE
4 PRINT#0,"U1:"4;0;TR;SE : REM
READ CONTENTS OF TRACK/SECTOR
INTO BUFFER
5 GET#4,TS,SS : REM READ FIRST 2
BYTES INTO BUFFER
6 TR=ASC(TS+CHR$(0)):
SE=ASC(SS+CHR$(0)) : REM MAKE
SURE VALUE IS INTEGER
7 IF TR=0THENCLOSE4;CLOSE0;END :
REM END OF LINKS
8 PRINT"TRACK NUMBER IS: ",TR,
"SECTOR NUMBER IS: ";SE
9 GOTO 4 : REM GET NEXT LINK

```

Fig. 2

```

1 OPEN#0,8,15 : REM OPEN COMMAND
CHANNEL
2 OPEN4,8,4,"#": REM OPEN DIRECT
ACCESS FILE
3 PRINT#0,"U1:"4;0;10;0 : REM
READ CONTENTS OF DESIRED
TRACK/SECTOR
4 PRINT#0,"B-P:"4;144 : REM POINT
TO WHERE YOU WANT TO READ FROM
5 FORX=1TO16 : REM LENGTH OF
FILENAME
6 GET#4,X$:IFX$=CHR$(16)THEN#0:
REM IF SHIFTED SPACE END
7 PRINTX$:NEXT : REM PRINT OUT
AND GET NEXT LETTER
8 CLOSE4;CLOSE0 : REM END

```

Fig. 3

would then have to read through all of the 143 bytes as the buffer ends you get to byte 144, the start of the name. However there is a quicker way. The B-P command allows you to point on the data pointer anywhere within the buffer. The bytes in the buffer are numbered from 0 to 255. The pointers automatically reset to 0 after a "U1" command. Figure 3 illustrates our example.

The commands block-write and block-read are used in conjunction with each other. As previously mentioned block-write allows you to write the contents of a buffer to a specified track and sector, the command does not alter the contents of the buffer - you do this yourself. Figure 4 takes the program in figure 2 and expands it so that the disk name read in can be stored in the buffer and then re-written to the correct position, changing the disk name.

When using Program, Sequential or Relative files on disk, the RAM is being constantly updated as programs are written, scratched, etc. This prevents programs from being overwritten. However, when we use direct-access files the data that you write to the disk is not marked in the BAM.

This means that data you have put on the disk could be overwritten. To prevent this from happening we can use the Block-Allocate command. If you try to allocate a block that has already been marked as used, then you will get an error message BS, NO BLOCK.TS : T and S are the next higher numbered free blocks available.

The syntax for using the block allocate command is:

B-A drive track sector

The following example would mark track 17 sector 5 as in use:

```

1 OPEN #0,13
2 PRINT #0,"B-A"017;5

```

Freeing a Block

The Block-Free of B-F command is the opposite of the above command. This will set the specified bits in the BAM making the relevant tracks and sectors available for use.

Should we want to free the sector allocated in the above example you would do it as following:

```

1 OPEN #0,13
2 PRINT #0,"B-F"017;5

```

```

1 OPEN0,8,15 : REM OPEN COMMAND
CHANNEL
2 OPEN4,8,4,"M" : REM OPEN DIRECT
ACCESS FILE
3 PRINT0,"U1:"4;0;10;0 : REM
READ CONTENTS OF DESIRED
TRACK/SECTOR
4 PRINT0,"B-P:"4;144 : REM POINT
TO WHERE YOU WANT TO READ FROM
5 X$="NEW DISK NAME"
6
IFLEN(X$)<16THENX$=X$+CHR$(160):
GOTO6 : REM PAD OUT NAME
7 PRINT04,X$ : REM CHANGE BUFFER
CONTENTS
8 PRINT0,"U2:"4;0;10;0 : REM
WRITE BACK TO DISK
9 PRINT0,"I":CLOSE4:CLOSE0:END

```

Fig. 4

Note—allocating and freeing blocks has an effect only on blocks that are used by PRG, SEQ and REL files by the DOS. The B-W and B-R commands do not check the BAM before overwriting blocks. Using these commands you can write to blocks marked as allocated in the BAM.

One use that has been made of this command in the past is to write a small menu program onto track 0, the directory track. This means that the MENU will not take up any of the normal disk space available.

Block-Escrow

The Block-Escrow command, shortened to B-E is an extremely powerful command. In essence, this command reads a sector from the disk into a previously opened buffer. The contents of that buffer are then executed as a machine code program within the buffer.

The syntax for the command is:
B-E channel drive track sector

When using the B-E command it is usual to specify the buffer to be used in the OPEN command, just in case the

machine code program isn't relocatable.

The following program would read the contents of track 14 sector 0 into buffer number 2 and execute it.

```

1 OPEN 0,0,15
2 OPEN 4,0,4"2"
3 PRINT 0,0,"B-E"400;144

```

Talking Memory

Not only are Commodore disk drives provided with a wealth of commands that allow you to access the disk but commands also exist that allow you to gain access to the memory inside the disk drive.

There are three commands that we will detail here. They are Memory Read (M-R), Memory Write (M-W), and Memory Escrow (M-E).

All of these commands require a knowledge of the inner workings of the DOS and a knowledge of address/byte. The memory map of the disk drive in figure 1 will be of invaluable use in this matter.

The syntax for the Memory Read command is:

```
M-R CHR$(LO) CHR$(HI)
```

[CHR\$(number)]

CHR\$(LO) is the low byte of the address in DOS that is to be read. CHR\$(HI) is the high byte of the address in DOS that is to be read. CHR\$(number) is an optional status parameter indicating how many bytes are to be read.

The figures 3 and 4 are used to illustrate the use of this command. The first example shows how to read from disk memory, how many bytes there are on the current disk. The second example reads the disk name.

Memory Write is the complimentary command to Memory Read. Writing can only be performed to DOS RAM, page zero, stack and buffers. It is possible to send more than one byte to the disk drive with this command. The syntax is as follows:

```
M-W CHR$(LO) CHR$(HI)
CHR$(number) CHR$(data)
CHR$(data) etc etc...
```

Finally the Memory Escrow command (M-E) will call and execute a machine code program that resides in DOS memory. The routine must end with a RTS instruction. The syntax for the command is as follows:

```
M-E CHR$(LO) CHR$(HI)
```

You can not only execute your own routine written with the use of the M-W command, but also the DOS routines.

Summary of Direct Access

Within the confines of this article I can obviously only gloss over the subject of programming your disk drive. The following table lists just a few ideas that spring to mind as tasks you could perform with your new-found knowledge.

- 1 You can manipulate the sectors and change the BAM
- 2 You can make changes to the directory.
- 3 You can make changes to files.
- 4 You can protect files from erasure.
- 5 You can close files that are OPENed.
- 6 You can prevent directories from being viewed.
- 7 You can recover lost or damaged files.
- 8 You can create data structures that the DOS would not normally recognize.

- 9 You could place a menu program within the directory - thus saving space.
- 10 You could put a simple form of protection on the disk.

Really, the list is endless. Only your own knowledge and requirements are your constraints. However, before you do try any of the commands that we have discussed out yourself, I must

stress the importance of making sure that you only play around with old disks until you know what you are doing. After all, one simple mistake could wipe out a whole disk.

Figure 1

```
1 OPEN0,0,10
2 PRINT0,"N-R"CHR$(150)CHR$(10)
3 GET0,X:IF0=""THEN0=CHR$(0)
4 PRINT0,"N-R"CHR$(150)CHR$(10)
5 GET0,Y:IF0=""THEN0=CHR$(0)
6 PRINT0("A01"=150*ABC(Y))
7 CLOSE0
```

Figure 2

```
1 OPEN0,0,10
2 PRINT0,"N-R"CHR$(144)CHR$(7)CHR$(10)
3 INPUT0,10
4 PRINT0
5 CLOSE0
```

DISK EDITING

You can rescue files and much more once you know how to use a disk editor.

By Stuart Cooke

How often have you scratched your latest programming masterpiece from your disk only to realize a few minutes later that you didn't have a backup?

No doubt, until now, the only option open to you was to re-type the whole program from the beginning.

A little more understanding about how a Commodore disk drive works will enable you to rescue most scratched programmes and make

numerous other changes to your disk directories.

Before you can start playing with the contents of floppy disks it is important that you understand how the information is stored on them. If you don't understand and you start changing areas of a disk you can probably cause bye-bye to the contents of the whole thing.

In order to make any changes to a disk you will require access to some

sort of disk editor program. There are a few available commercially, Disk Doctor from Precision being a good example, and we provide a listing for a good disk editor later in the supplement.

Disk Structure

You are no doubt aware, when you purchase a disk it is totally blank and of no use to you at all. Before the

Block Distribution By Track

Track Numbers		Range of Sectors		Total Sectors		Single Sided	Double Sided
HEX	DEC	HEX	DEC	HEX	DEC		
#01-#11	01-17	#00-#14	00-20	#16	21	YES	YES
#12-#18	18-24	#00-#12	00-18	#18	18	YES	YES
#19-#25	25-30	#00-#11	00-17	#12	15	YES	YES
#1F-#2D	31-36	#00-#10	00-16	#11	17	YES	YES
#24-#34	38-52	#00-#14	00-20	#15	21	NO	YES
#35-#3B	53-59	#00-#12	00-18	#12	15	NO	YES
#3C-#41	60-65	#00-#11	00-17	#12	15	NO	YES
#42-#48	66-72	#00-#10	00-16	#11	17	NO	YES

Fig. 1

computer/disk drive can make use of the disk it must be formatted.

Formatting a disk divides it into a number of rings-called tracks. On all of the popular Commodore drives (1541 and so) except for the 1571, the disk is divided into 35 rings, on one side of the disk only. If you have a 1571 then the second side of the disk is also split into 35 rings-or tracks.

Each of these circular tracks is then split up into a number of equal segments called sectors. Each track contains between 17 and 21 sectors. Figure 1 illustrates this more clearly. Note that the tracks on the second side of a 1571 are numbered from 36 to 70 and do not start from 1 again.

As Figure 1 clearly shows the number of sectors in each track goes smaller towards the center of the disk. The reason for this is quite obvious when you realize that the tracks are a lot shorter at the center of the disk than they are on the outside.

How much room?

In the center of the disk, side 1 for 1571 users, you will find the information track. Track 18 is used to keep all necessary information about programmes, where they are stored on the disk and how much room is free on the disk.

The first sector of track 18 is used to record which sectors of the disk have been used. This is called the Block Availability Map or BAM. Every time you make any changes to the contents of a disk the contents of

the BAM are updated so the disk drive can find out which tracks and sectors on the drive are used.

Figure 2 shows the contents of the first 255 bytes of track 18 sector 0. As you can see from the figure this sector not only contains information about the BAM but is also used to store important information about the disk, such as the DOS type, the format type, the name, etc.

Returning to the BAM, figure 2 shows that bytes 4 to 143 of track 18 sector 0 holds the BAM. For consecutive bytes are used to represent the BAM for each track. Figure 3 gives a representation of the possible contents of sector 0 bytes 5 to 7, in other words the bytes that give an indication of which sectors on track 1 have been used.

As you are no doubt aware, a single byte can hold a number from 0 to 255. If we translate this from decimal to binary this means that the numbers held will range from 00000000 to 11111111. From the binary representation it can be seen that each byte can hold the information for eight sectors. Each 1 or 0 represents the status of the corresponding sector. A 0 tells the disk drive that the sector in question has been used while a 1 shows that it is still available.

If you take a second look at Figure 3 you can see that in our representation sectors 1 to 16 have all been used and sectors 17 to 23 are still available.

You may be wondering how the disk drive knows how many sectors are

available on each track. If you refer back to Figure 2 you will see that the information about the BAM for each track is held in four consecutive bytes. We have already taken a close look at the BAM for track 1 above. As stated this information is stored in bytes 5 to 7 of track 18 sector 0. If you refer back to Figure 2 you will see that the previous byte (4) holds a number that represents the actual number of sectors available on track 1, in this case 23.

This sequence of four bytes is repeated for all tracks on the disk. 1571 users can see that the information about the second side of the disk is stored in the same way as the first side in bytes 221 to 255 of track 18 sector 0.

Disk Info

Bytes 144 to 255 of track 18 sector 0 are used to hold specific information about the disk. Much of this is information that is printed at the top of each directory listing. If you refer back to Figure 2 you will see exactly what information is held in these bytes. Should you ever want to change the ID or the title of a disk then you can do it quite simply by using a disk editor to read the information on the disk into your computer, make the changes required and then re-write the information to the disk.

Directory Info

The sectors from one onwards on track 18 are used to hold information

BAM Format 1541 - Track 18 Sector 0

Number	Contents	Definition
0	18	Track of next directory block. Always 18.
1	1	Sector of next directory block Always 1.
2	65	ASCII character A indicating 1541/81/71/4040 format
3		Double sided flag. Ignored on 1541
4		Number of sectors available on track 1.
5		Track 1, Sector 0-7 BAM.
6		Track 1, Sector 8-16 BAM.
7		Track 1, Sector 17-23 BAM.
8		Number of sectors available on track 2.
9		Track 2, Sector 0-7 BAM.
10		Track 2, Sector 8-16 BAM.
11		Track 2, Sector 17-23 BAM.
...	etc. Down To ...	
140		Number of sectors available on track 25.
141		Track 25, Sector 0-7 BAM.
142		Track 25, Sector 8-16 BAM.
143		Track 25, Sector 17-23 BAM.
144-159		Disk name padded with shifted spaces (CHR# 1001).
160-161	160	Shifted space.
162-163		Disk ID.
164	160	Shifted space.
165-166		ASCII "A" which is the DOS version format type 1540/41/81/71/4040.
167-170	160	Shifted spaces.
171-255	0	Nulls, not used.

1571 Drive As Above Except :

0		Double sided flag: #00=Double Sided #01=Single Sided.
171-255	0	Nulls, not used.
221-227		Number of sectors available in tracks 26-32.
		Each sector by each byte.
		Format as for 1541.
228	0	Number of sectors in track 33
229-244		Number of sectors available tracks 34-38. Each track by each byte.
245-250		Number of sectors available tracks 39-43.
251-255		Number of sectors available tracks 44-70.

RAM ALLOCATION		
SECTORS 0-7	SECTORS 8-15	SECTORS 16-23
00000000	00011111	11111111

Fig. 3

relating to any program you have stored on the disk. Each sector is referred to as a *directory block* and will hold the information for around eight files. The first two bytes of each block are used to give the track and sector of the next directory block. Figure 4 shows the format of the directories on the disk. If there is no more information on the disk, the first two

bytes in the last sector will contain 0's. Each of the eight program entries in a directory block is made up of 30 bytes. These are the ones that hold the information about the type of program, where it is on the disk, etc. Figure 4 shows what information stored in the 30 bytes.

The first byte of each directory entry is used to hold information

about the type of file that you are looking at. If you refer to figure 4 once more you will see that the file referred to can be one of five types. However, this isn't the only information that this byte gives.

Bits 0 to 2 of this byte are used to tell us what type of file we are looking at. Bit 7 is used to tell the drive if the file is correctly closed or not. A 1 in bit number 7 shows that the file is still open. This can be seen on a directory listing as a '*' following the filetype.

Bit 6 holds an extremely important piece of information which, unfortunately, a large number of people are unaware of. This bit is used to tell the disk drive whether or not the file is protected. Setting this bit to '1' will prevent deleting the file by normal methods. A protected file can be seen in a directory listing as it has a 'p' following the filetype. If you have important files it is well worth going to the trouble of setting this bit to prevent accidental erasure.

Program Erasure

Whenever you delete a program from disk a number of changes are made to the disk. First of all, the sectors that the program occupied are marked as free in the RAM and secondly, the file type is changed to one indicating that it has been deleted. The important thing to remember is that the program is still on the disk and will remain there until another program is saved over it, probably following the next SAVE operation.

If you delete a file by accident and realise before you have saved another to disk that it is a very simple matter to retrieve it. All you have to do is find the entry for the file in the directory block and change the filetype to whatever it was before. If, for example, the file type was a program you would change the number 02 to the relevant position. You will now be able to use your file.

Note, the RAM will not be updated and there is a chance that the next SAVE operation may overwrite your restricted file. It is therefore a good idea to make a new copy of any restricted file before doing anything else.

Having taken a look at the way that a disk directory is stored on a Commodore disk it is probably worth looking at the format that files take. Figures 5 to 7 give details on all of

Dir File Format, Track 18 Sectors 1-19

Byte	Definition
0-1	Track and sector of next DIR block.
2-21	File Entry 1
24-63	File Entry 2
66-95	File Entry 3
98-127	File Entry 4
130-159	File Entry 5
162-191	File Entry 6
194-223	File Entry 7
226-255	File Entry 8

Structure Of Each Directory Entry

Byte	Contents	Definition
0	DIR-type	File type 08'd with 000 to indicate closed file.
		File type 08'd with 000 to indicate locked file. Types: 0 = Deleted. 1 = Sequential. 2 = Program. 3 = User. 4 = Relative.
1-2		Track and sector of first data block.
3-14		File name padded with shifted spaces.
15-23		REL File only. Track and sector of last data sector.
24		REL File only. Record length.
25-26		STATUS.
27-29		Track and sector of replacement during REPLACE or SUPER.
30-39		Number of blocks in file, stored as a two-byte integer in normal 16-byte 31-byte format.

Fig. 4

Relative File Format

Byte Definition

DATA BLOCK

0,1 Track and Sector of next data block.
 2-255 254 bytes of data. Empty Records contain \$FF in the first byte followed by \$00 to the end of the record. Partially filled records are padded with \$00.

SIDE SECTOR BLOCK

0,1 Track and Sector of next data block.
 2 Side sector number (0-5).
 3 record length.
 4-5 Track and Sector of 1st side sector (0).
 6-7 Track and Sector of 2nd side sector (1).
 8-9 Track and Sector of 3rd side sector (2).
 10-11 Track and Sector of 4th side sector (3).
 12-13 Track and Sector of 5th side sector (4).
 14-15 Track and Sector of 6th side sector (5).
 16-255 Track and sector pointers to 120 data blocks.

Fig. 7

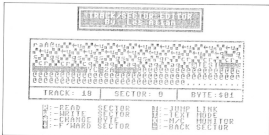


Fig. 8

Firstly, you will need to format a blank disk. Make sure that it is blank and contains nothing that you require before going any further.

Put your disk in your drive and enter the following command:

```
OPEN8,15,"NO TEST.FC"
```

Next type the following small program and SAVE it onto your disk with the filename "ONE"

```
10 REM THIS IS A VERY
20 REM SHORT TEST
30 REM PROGRAM
40 REM
50 REM THE END
```

OK, so it's nothing stunning but it will serve our purpose very well.

Now LOAD your disk editor and examine the contents of track 15 sector 0. If you have a look at bytes 144 to 161 you will see that they hold the name of the disk. Figure 8 shows what you would see using our disk editor, the

display may be slightly different with your editor.

Now we shall change the disk name. Change the letters of the filename to "DEMO". Figure 9 shows how your disk should look now.

Once you have done this write the sector back to the disk and your changes will have been made permanent. If you want to check this for yourself reset your machine and load in the disk directory, you will see that the name has been changed.

Remove a file

Now we are going to delete a file and then recover it. Scratch the test file from your disk with the following command:

```
OPEN8,15,"NO ONE"
```

If you now try and LOAD the program "ONE" you will be unable to

do so.

Load your disk editor into your computer and take a look at track 15 sector 1. Figure 10 shows something similar to what you should see. The 00 byte indicated on our picture shows that the file has been deleted. If you use the editor to change this to 82, it is a program file, and then write the sector back to disk you will be able to LOAD the program once again.

Obviously this article has only glossed over the area of disk structure and disk editing. If you have old disks that you no longer want don't format them straight away, play around with them using a disk editor till you feel sure that you know exactly what you are doing.

REMEMBER never edit a disk that has programmes on it that you require unless you are sure about what you're doing.



Fig. 8

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
.....																																																																																																																																																																																																																																																															
TRACK: 15 SECTOR: 0 BYTE: 780																																																																																																																																																																																																																																																															

```
N - READ    SECTOR
M - WRITE   SECTOR
M - F*WARD SECTOR
```

```
N - JUMP    LINE
M - TEXT    LINE
M - R*U*E   LINE
M - S*O*U*E SECTOR
```

Fig. 10



00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
.....																																																																																																																																																																																																																																																															
TRACK: 15 SECTOR: 1 BYTE: 780																																																																																																																																																																																																																																																															

```
N - READ    SECTOR
M - WRITE   SECTOR
M - F*WARD SECTOR
```

```
N - JUMP    LINE
M - TEXT    LINE
M - R*U*E   LINE
M - S*O*U*E SECTOR
```

Track/Sector Editor for CBM 64/ 128

Hints on how to use the editor

By Les Allan

It is often useful to edit a section of memory either resident in the computer's memory or, as explained in this utility, by modifying the saved file directly on disk. It is extremely important that a "backup" be made prior to making any alterations, so in the event of a mistake the user can always revert back to the original.

Boot up the TRACK/SECTOR EDITOR and you are directly into the READ SECTOR mode. Insert the disk to be read and simply press RETURN twice to select the BAM of the directory (default mode is TRACK 18 and SECTOR 0) or input the required TRACK and SECTOR as commanded by the cursor. The contents of that sector are now displayed in the viewing window, with the cursor flashing at the top left hand corner - position 0. The first two bytes are highlighted in white to indicate the LINK to the location of the next track and sector of the saved file.

T - TEXT MODE

Use the cursor control keys to locate the cursor to the required position and PRESS **T** to enter TEXT mode. Simply type in from the keyboard the text required using the CTRL key to select lower case and press RETURN to end.

+ = CHANGE BYTE

Use the cursor control keys to locate the cursor to the required position and

PRESS the ***** key. The value of the byte to be changed can be entered either directly in decimal or in hex provided the number is preceded by the **H** key.

W - WRITE SECTOR

To write the modified sector to the disk PRESS the **W** key and confirm your intention by pressing the **T** key or quit with **N** key.

R - READ SECTOR

PRESS the **R** key at any time to select a

A	C000	LDA SC000	assemble code at SC000
D	C000		disassemble code at SC000
M	C000 C100		monitor code between SC000 and SC100
F	C000 C100 EE		fill with EE between SC000 and SC100
S	"SECTOR",08,C000,C100		save contents of sector to disk
L	"SECTOR",08		load sector back to SC000
N			quit monitor and return to the editor

different track and sector to be read.

F1 - RE-START

PRESS the F1 key at any time during the edit mode to reset the registers to the start up configuration of TRACK 18 and SECTOR 0.

J - VIEW NEXT SECTOR

PRESS the HOME key to position the cursor to the LINK track and sector followed by the J key will cause the editor to jump to the next sector of the saved file.

K - M/C MONITOR

Press the K key to enter the M/C monitor which will display the contents of the sector between SCOR0 and SC100.

Figure 1 gives the available commands.

Directory Entry Explanation

Boot up the TRACK/SECTOR EDITOR, insert disks to be read, press RETURN twice to read RAM and J to jump to TRACK 18 and SECTOR 1 which is always the first sector of the directory. Figure 2 gives details.

Up to eight directory entries are saved per sector in exactly the same sequence. To view the contents of any saved file simply locate the sector at byte 4 and press J to read the first track and sector.

byte 1	312 location of next track of directory - 500 if end
byte 2	504 location of next sector of directory - 5FF if end
byte 3	502 type of file saved (PRG) - change to 5C2 to LOCK file
byte 4	501 start track of saved file
byte 5	500 start sector of saved file
byte 6 - 21	NAME of save file
byte 22 - 26	relative file data
byte 31	length of saved file in bytes
byte 32	000

Fig. 1

PROGRAM: disk editor boot

32	18 00000000000000000000000000000000	32	20 00000000000000000000000000000000	16	00 data 01 02 03 04 05 06 07
33	18 00000000000000000000000000000000	33	20 00000000000000000000000000000000	16	08 08 08 08 08 08 08 08 08 08 08 08
34	18 00000000000000000000000000000000	34	20 00000000000000000000000000000000	16	09 09 09 09 09 09 09 09 09 09 09 09
35	18 00000000000000000000000000000000	35	20 00000000000000000000000000000000	16	0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
36	18 00000000000000000000000000000000	36	20 00000000000000000000000000000000	16	0B 0B 0B 0B 0B 0B 0B 0B 0B 0B 0B 0B
37	18 00000000000000000000000000000000	37	20 00000000000000000000000000000000	16	0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C
38	18 00000000000000000000000000000000	38	20 00000000000000000000000000000000	16	0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
39	18 00000000000000000000000000000000	39	20 00000000000000000000000000000000	16	0E 0E 0E 0E 0E 0E 0E 0E 0E 0E 0E 0E
40	18 00000000000000000000000000000000	40	20 00000000000000000000000000000000	16	0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
41	18 00000000000000000000000000000000	41	20 00000000000000000000000000000000	16	10 10 10 10 10 10 10 10 10 10 10 10
42	18 00000000000000000000000000000000	42	20 00000000000000000000000000000000	16	11 11 11 11 11 11 11 11 11 11 11 11
43	18 00000000000000000000000000000000	43	20 00000000000000000000000000000000	16	12 12 12 12 12 12 12 12 12 12 12 12
44	18 00000000000000000000000000000000	44	20 00000000000000000000000000000000	16	13 13 13 13 13 13 13 13 13 13 13 13
45	18 00000000000000000000000000000000	45	20 00000000000000000000000000000000	16	14 14 14 14 14 14 14 14 14 14 14 14
46	18 00000000000000000000000000000000	46	20 00000000000000000000000000000000	16	15 15 15 15 15 15 15 15 15 15 15 15
47	18 00000000000000000000000000000000	47	20 00000000000000000000000000000000	16	16 16 16 16 16 16 16 16 16 16 16 16
48	18 00000000000000000000000000000000	48	20 00000000000000000000000000000000	16	17 17 17 17 17 17 17 17 17 17 17 17
49	18 00000000000000000000000000000000	49	20 00000000000000000000000000000000	16	18 18 18 18 18 18 18 18 18 18 18 18
50	18 00000000000000000000000000000000	50	20 00000000000000000000000000000000	16	19 19 19 19 19 19 19 19 19 19 19 19
51	18 00000000000000000000000000000000	51	20 00000000000000000000000000000000	16	1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A
52	18 00000000000000000000000000000000	52	20 00000000000000000000000000000000	16	1B 1B 1B 1B 1B 1B 1B 1B 1B 1B 1B 1B
53	18 00000000000000000000000000000000	53	20 00000000000000000000000000000000	16	1C 1C 1C 1C 1C 1C 1C 1C 1C 1C 1C 1C
54	18 00000000000000000000000000000000	54	20 00000000000000000000000000000000	16	1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D 1D
55	18 00000000000000000000000000000000	55	20 00000000000000000000000000000000	16	1E 1E 1E 1E 1E 1E 1E 1E 1E 1E 1E 1E
56	18 00000000000000000000000000000000	56	20 00000000000000000000000000000000	16	1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F
57	18 00000000000000000000000000000000	57	20 00000000000000000000000000000000	16	20 20 20 20 20 20 20 20 20 20 20 20
58	18 00000000000000000000000000000000	58	20 00000000000000000000000000000000	16	21 21 21 21 21 21 21 21 21 21 21 21
59	18 00000000000000000000000000000000	59	20 00000000000000000000000000000000	16	22 22 22 22 22 22 22 22 22 22 22 22
60	18 00000000000000000000000000000000	60	20 00000000000000000000000000000000	16	23 23 23 23 23 23 23 23 23 23 23 23
61	18 00000000000000000000000000000000	61	20 00000000000000000000000000000000	16	24 24 24 24 24 24 24 24 24 24 24 24
62	18 00000000000000000000000000000000	62	20 00000000000000000000000000000000	16	25 25 25 25 25 25 25 25 25 25 25 25
63	18 00000000000000000000000000000000	63	20 00000000000000000000000000000000	16	26 26 26 26 26 26 26 26 26 26 26 26
64	18 00000000000000000000000000000000	64	20 00000000000000000000000000000000	16	27 27 27 27 27 27 27 27 27 27 27 27
65	18 00000000000000000000000000000000	65	20 00000000000000000000000000000000	16	28 28 28 28 28 28 28 28 28 28 28 28
66	18 00000000000000000000000000000000	66	20 00000000000000000000000000000000	16	29 29 29 29 29 29 29 29 29 29 29 29
67	18 00000000000000000000000000000000	67	20 00000000000000000000000000000000	16	2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
68	18 00000000000000000000000000000000	68	20 00000000000000000000000000000000	16	2B 2B 2B 2B 2B 2B 2B 2B 2B 2B 2B 2B
69	18 00000000000000000000000000000000	69	20 00000000000000000000000000000000	16	2C 2C 2C 2C 2C 2C 2C 2C 2C 2C 2C 2C
70	18 00000000000000000000000000000000	70	20 00000000000000000000000000000000	16	2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
71	18 00000000000000000000000000000000	71	20 00000000000000000000000000000000	16	2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E
72	18 00000000000000000000000000000000	72	20 00000000000000000000000000000000	16	2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F
73	18 00000000000000000000000000000000	73	20 00000000000000000000000000000000	16	30 30 30 30 30 30 30 30 30 30 30 30
74	18 00000000000000000000000000000000	74	20 00000000000000000000000000000000	16	31 31 31 31 31 31 31 31 31 31 31 31
75	18 00000000000000000000000000000000	75	20 00000000000000000000000000000000	16	32 32 32 32 32 32 32 32 32 32 32 32
76	18 00000000000000000000000000000000	76	20 00000000000000000000000000000000	16	33 33 33 33 33 33 33 33 33 33 33 33
77	18 00000000000000000000000000000000	77	20 00000000000000000000000000000000	16	34 34 34 34 34 34 34 34 34 34 34 34
78	18 00000000000000000000000000000000	78	20 00000000000000000000000000000000	16	35 35 35 35 35 35 35 35 35 35 35 35
79	18 00000000000000000000000000000000	79	20 00000000000000000000000000000000	16	36 36 36 36 36 36 36 36 36 36 36 36
80	18 00000000000000000000000000000000	80	20 00000000000000000000000000000000	16	37 37 37 37 37 37 37 37 37 37 37 37
81	18 00000000000000000000000000000000	81	20 00000000000000000000000000000000	16	38 38 38 38 38 38 38 38 38 38 38 38
82	18 00000000000000000000000000000000	82	20 00000000000000000000000000000000	16	39 39 39 39 39 39 39 39 39 39 39 39
83	18 00000000000000000000000000000000	83	20 00000000000000000000000000000000	16	3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A
84	18 00000000000000000000000000000000	84	20 00000000000000000000000000000000	16	3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B
85	18 00000000000000000000000000000000	85	20 00000000000000000000000000000000	16	3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C
86	18 00000000000000000000000000000000	86	20 00000000000000000000000000000000	16	3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D
87	18 00000000000000000000000000000000	87	20 00000000000000000000000000000000	16	3E 3E 3E 3E 3E 3E 3E 3E 3E 3E 3E 3E
88	18 00000000000000000000000000000000	88	20 00000000000000000000000000000000	16	3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F
89	18 00000000000000000000000000000000	89	20 00000000000000000000000000000000	16	40 40 40 40 40 40 40 40 40 40 40 40
90	18 00000000000000000000000000000000	90	20 00000000000000000000000000000000	16	41 41 41 41 41 41 41 41 41 41 41 41
91	18 00000000000000000000000000000000	91	20 00000000000000000000000000000000	16	42 42 42 42 42 42 42 42 42 42 42 42
92	18 00000000000000000000000000000000	92	20 00000000000000000000000000000000	16	43 43 43 43 43 43 43 43 43 43 43 43
93	18 00000000000000000000000000000000	93	20 00000000000000000000000000000000	16	44 44 44 44 44 44 44 44 44 44 44 44
94	18 00000000000000000000000000000000	94	20 00000000000000000000000000000000	16	45 45 45 45 45 45 45 45 45 45 45 45
95	18 00000000000000000000000000000000	95	20 00000000000000000000000000000000	16	46 46 46 46 46 46 46 46 46 46 46 46
96	18 00000000000000000000000000000000	96	20 00000000000000000000000000000000	16	47 47 47 47 47 47 47 47 47 47 47 47
97	18 00000000000000000000000000000000	97	20 00000000000000000000000000000000	16	48 48 48 48 48 48 48 48 48 48 48 48
98	18 00000000000000000000000000000000	98	20 00000000000000000000000000000000	16	49 49 49 49 49 49 49 49 49 49 49 49
99	18 00000000000000000000000000000000	99	20 00000000000000000000000000000000	16	4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A 4A
100	18 00000000000000000000000000000000	100	20 00000000000000000000000000000000	16	4B 4B 4B 4B 4B 4B 4B 4B 4B 4B 4B 4B

DIR Cover

It can sometimes be a real pain when trying to find one specific program that's hidden away on one of over 100 disks. DIR COVER will make life much easier as it will produce, a disk cover on a printer, that you can cut out and make. A total list of all the programs found on the disk will be printed on the cover, together with lots more useful information about the programs.

By Elizabeth Mackenzie

Dir cover is designed for the Gemini 10X printer, although slight modifications can be made for other printers. The program is simple to use, as it's just a case of answering the prompts. It will not accept wrong answers (with this reason, though it will accept anything between 01-12 for the month or 01-31 for the day), when entering the date.

If you choose the first prompt, (to screen) then you may view the directory as it will be printed before choosing your print options. As well as directory or plain disk covers, you can now print a directory to the right or left column, (saving paper) or in double width, if it is only list type which are needed, (with or without the start addresses or just names).

It is very handy (and can save hair pulling) to have this as well as the usual information, plus track, sector, date,

disk name and number, and also blocks free on the cover.

Further information can be added, when saving a program to disk. You press shift/space after the filename. Then type one or more letters to indicate the information you may need, before typing the closing quotes. For example:

SAVE"MC:PROGRAMS/SP15".

When the directory is listed the '15' is outside the quotes, you do not include this with the filename when loading, it just tells you to load with .R1 and SYS, the SYS number being normally the start address which could be on the cover.

Print the covers on different coloured paper and you'll have a rainbow library (as I have), with a different colour for different types of files. Also you can see at a glance which disks are games or utilities, etc.

If you do have printing problems, refer to your printer manual. The lines that may need altering are: (L38 REMOVE RIM), (L39 and 308 TOGGLE PRINT). You could remove the REM from line 1650, (L166, 159 and 2040 2050 SET LEFT MARGIN). For someone with a non-graphic printer, the graphics could be changed in lines 570-640, 2050-2060, 2350-2360. The other graphic lines are obvious. Use '!' and the dash '-'.

The 87 in line 418 will need to be changed each year if you don't want a tack answer.

The 75 in lines 2350, 2470, 2480, can be altered if you have more than 75 files on your disk, and it will depend on paper length, (you can get it cut to the length you want). More files will then be printed to the back flap and can be turned in. Faring the sides of the flap allows easy access.


```

14 2520 print@6:and(peek(1)"/line
15 2521 or:and(peek2,peek(1)"/or
20 2530 for:for(20)pr:pr1844:and:
21 2531 and:and(1)
22 2540 peek@:700:peek:1:and
23 2550 for:for(10)pr:print@:and:
24 2551 and:and(1)
25 2560
26 2570
27 2580
28 2590
29 2600
30 2610
31 2620
32 2630
33 2640
34 2650
35 2660
36 2670
37 2680
38 2690
39 2700
40 2710
41 2720
42 2730
43 2740
44 2750
45 2760
46 2770
47 2780
48 2790
49 2800
50 2810
51 2820
52 2830
53 2840
54 2850
55 2860
56 2870
57 2880
58 2890
59 2900
60 2910
61 2920
62 2930
63 2940
64 2950
65 2960
66 2970
67 2980
68 2990
69 3000
70 3010
71 3020
72 3030
73 3040
74 3050
75 3060
76 3070
77 3080
78 3090
79 3100
80 3110
81 3120
82 3130
83 3140
84 3150
85 3160
86 3170
87 3180
88 3190
89 3200
90 3210
91 3220
92 3230
93 3240
94 3250
95 3260
96 3270
97 3280
98 3290
99 3300

```

1541 Fast Loader

The 1541 disk drive has been described as the ' lumbering Hippo' of disk drives. Speed it up with this fast loader.

By Paul Eves

The Commodore 1541 disk drive is notorious for being one of the slowest disk drives available for any computer. It may seem silly that some of the earliest file loading systems for the C64 are actually faster than this disk drive.

The fast loader program presented here patches itself into the C64's memory and improves on the speed of loading.

All that you need to do is LOAD and RUN the program "FAST LOADER" and the changes to disk loading speed will become very

obvious.

It is worth pointing out at this stage that the fast loader does occupy some of the C64's memory. It is therefore possible for some programmes to corrupt the fast loader preventing it from working.

Getting it all in

The program is presented here as a Basic loader and should be typed in as a normal Basic program. When you have finished typing it in save it to disk with the name "FAST LOAD.BAS".

Type the following line and press RETURN

```
POKE 450,POKE 64,16NEW
```

Now LOAD the program "FAST LOAD.BAS" from your disk and RUN it.

When you want to use the fast loader simply load it into your C64 and RUN it. You will find when it is patched into the computers operating system.

C64 UTILITY

```

Getting it in          4f 6 rem * poke's before      *
FAST LOADER          56 7 rem * loading and running *
1) Type in the BASIC program
presented here.
2) SAVE the program onto disc.
3) Type NEW.
4) Enter the following:
    POKE 40,0:POKE 44,10:NEW
5) 20 for 1=0 to bl:cx=0:for d=0
    to 15:read a:ccccxta:poke aa
    +1*15+d,a:next d
6) LOAD and RUN the program saved
in 2.
7) When finished enter the
following to SAVE the program :
    POKE40,1:POKE44,8:POKE48,102:
    POKE49,13:SAVE"FASTLOAD".8
8) The program will now be on
disc.

6a 80 data 33,32,169,6,141,33,20
8,163,0,189,14,8,240,6,32,210
,1482
6b 90 data 255,232,208,245,120,1
68,0,132,251,169,224,139,252,
177,251,145,2954
6c 100 data 251,200,208,249,230,
252,200,245,169,249,133,252,1
69,191,183,0,3175
6d 110 data 139,253,134,254,177,
253,145,251,200,208,249,230,2
54,230,252,165,3088
6e 120 data 252,201,252,144,239,
169,239,141,214,253,162,34,19
9,158,0,157,2800
6f 130 data 193,2,202,16,247,32,
191,8,141,76,253,142,77,253,1
69,219,2239
6g 140 data 162,2,141,35,239,142
,40,229,88,96,0,72,169,53,133
,1,1592
6h 150 data 104,30,111,240,72,16
9,72,141,143,2,169,235,141,14
4,2,169,1294
6i 160 data 25,133,1,104,96,0,16
9,53,133,1,76,72,235,0,169,19
2,1489
6j 170 data 163,2,141,48,9,142,4
9,3,96,120,169,39,141,0,221,4
4,1380
6k 180 data 0,221,80,251,169,3,1
41,0,221,162,9,202,200,253,16

```

PROGRAM: FAST LOAD.BAS

	340 data 173.0,221.10,8,10,38 .251,40,38,251,203,200,243,18 1,251,2124		
57	200 data 145,174,200,208,233, 169,23,141,0,221,165,251,98,1 20,169,29,2304	04	300 data 41,239,141,17,208,16 9,8,32,12,237,169,111,32,183, 237,169,2007
eb	210 data 141,0,221,44,0,221,8 0,251,149,3,141,0,221,143,8,2 02,1864	04	390 data 77,32,221,237,169,45 .32,221,237,169,69,32,221,237 ,169,3,2171
36	220 data 208,253,162,4,173,0, 221,10,8,10,38,251,40,38,251, 202,1869	17	400 data 32,221,237,169,3,32, 221,237,32,254,237,169,7,141, 0,221,2213
67	230 data 208,242,169,23,141,0 .221,236,234,234,169,251,98,1 33,147,169,2667	36	410 data 162,0,202,208,253,19 4,253,32,63,240,201,255,240,9 0,160,2,2503
64	240 data 0,133,144,165,166,20 1,8,240,3,76,171,244,164,183, 208,3,2129	38	420 data 166,253,208,23,72,32 .63,240,169,32,63,248,168,183 ,208,4,2139
8e	250 data 76,16,247,140,230,25 1,140,0,177,107,153,201,251,1 92,0,208,2519	5e	430 data 164,195,165,196,132, 174,133,175,160,4,104,203,0,2 40,20,132,2185
d6	260 data 4,201,26,240,228,200 .196,183,144,238,32,175,245,1 73,24,3,2322	84	440 data 253,56,165,174,229,2 53,133,174,176,3,198,195,32,1 1,248,230,2509
dc	270 data 72,173,25,3,72,169,1 93,162,254,141,24,3,142,25,3, 169,1630	77	450 data 175,208,198,32,63,24 8,133,253,160,0,165,253,201,2 .144,10,2343
e0	280 data 130,141,13,221,169,1 .141,6,221,169,0,141,7,221,16 9,25,1775	ab	460 data 32,63,248,145,174,20 0,198,253,208,240,169,289,133 ,253,152,24,2747
29	290 data 141,15,221,169,8,141 .15,221,104,141,25,3,104,141, 24,3,1476	85	470 data 101,174,133,174,144, 2,230,175,173,173,208,9,16,141 .17,208,1922
83	300 data 173,21,208,133,254,1 69,0,141,21,208,169,19,162,25 0,133,3,2064	32	480 data 165,254,141,21,200,1 73,13,221,169,127,141,13,221, 88,165,253,2373
fe	310 data 134,4,162,0,169,3,13 4,3,133,6,169,8,32,12,237,169 .1377	b7	490 data 208,3,76,4,247,201,1 28,208,3,76,7,247,76,169,245, 0,1098
2b	320 data 111,32,185,237,169,1 44,10,7,169,128,133,253,76,22 0,249,169,2294	8f	500 data 0,76,8,4,169,8,141,0 .24,76,126,3,162,1,88,138,102 4
09	330 data 77,32,221,237,169,45 .32,221,237,169,97,32,221,237 .163,5,2177	8f	510 data 44,0,24,240,251,120, 189,0,141,0,24,138,44,0,24,20 0,1427
6b	340 data 32,221,237,165,6,32, 221,237,169,29,32,221,237,160 .0,177,2176	f9	520 data 251,234,162,4,177,10 .73,255,133,20,169,0,6,20,42, 10,1546
77	350 data 3,32,221,237,200,192 .29,144,246,32,254,237,24,165 .3,165,2124	b4	530 data 4,20,42,10,141,0,24, 202,208,240,234,234,234,200,2 08,226,2223
bd	360 data 29,133,3,144,3,230,4 .24,165,9,168,6,205,29,133,3, 1194	02	540 data 234,234,234,169,8,14 1,0,24,98,73,253,88,133,20,16 2,1,1872
02	370 data 144,3,233,230,6,234, 4,144,161,201,228,144,157,173	2a	550 data 130,44,0,24,240,251, 120,169,0,141,0,24,138,44,0,2 4,1357
		f7	560 data 208,251,162,4,169,0, 6,28,42,10,8,20,42,30,141,0,1

	091		1405
74	570 data 24,202,208,240,182,3 .302,308,353,169,8,141,0,24,9 6,32,1972	45	690 data 14,208,18,230,14,173 .2,6,32,72,3,173,9,8,32,72,10 58
09	580 data 24,193,169,0,182,8,1 33,18,134,11,133,14,169,8,133 .249,1546	47	700 data 3,160,4,208,2,160,2, 185,6,240,11,32,11,3,173,1,11 81
fa	590 data 169,2,133,106,169,18 .133,6,169,1,133,7,32,119,4,1 60,1361	1a	710 data 8,133,7,78,29,4,173, 1,6,32,72,3,136,204,1,6,809
24	600 data 35,201,1,208,80,180, 0,185,2,6,41,135,201,130,208, 53,1646	49	720 data 176,10,200,185,0,6,3 2,72,3,76,92,4,169,0,141,0,11 58
46	610 data 162,0,240,36,189,212 .4,217,5,6,240,11,201,63,208, 37,1823	43	730 data 24,169,1,133,28,78,1 48,183,182,0,134,15,134,12,18 6,28,1423
cc	620 data 188,5,8,201,160,240, 30,232,200,236,211,4,178,9,18 9,212,2296	fc	740 data 340,9,169,0,133,28,1 69,176,32,189,4,169,224,32,18 9,4,1767
04	630 data 4,201,42,240,59,208, 221,152,41,31,201,16,176,50,1 85,5,1832	b0	750 data 201,3,200,41,165,13, 208,37,230,12,169,192,32,189, 4,169,1871
45	640 data 6,201,180,240,43,152 .41,224,24,165,32,168,144,185 .173,0,1890	bd	760 data 176,32,189,4,201,1,2 08,21,78,138,4,201,3,208,14,1 65,1841
16	650 data 6,208,14,160,98,169, 255,32,72,3,169,0,141,0,24,15 2,1505	15	770 data 15,208,10,230,15,169 .192,32,189,4,76,138,4,96,141 .91,1610
5b	660 data 78,200,193,173,1,6,7 6,153,3,169,6,133,49,78,209,2 44,1767	10	780 data 2,141,77,2,133,0,169 .255,141,152,2,182,0,88,32,16 6,1522
dc	670 data 152,41,224,188,185,3 .6,133,6,185,4,6,129,7,32,119 .1404	73	790 data 213,176,251,96,165,3 53,208,3,76,4,247,201,128,208 .3,76,2308
51	680 data 4,180,35,201,1,208,2 06,173,0,6,133,6,32,72,3,169,	e4	800 data 7,247,76,169,345,73, 1,169,245,0,0,0,0,255,0,0,148 7

Menu Maker

Make the loading and running of files much easier with this handy menu program.

By Tony Crowther

When loading a program from disk it can sometimes be quite difficult to remember exactly how a program should be loaded. Three months after writing your allowing, all-dancing utility, the chances of you remembering whether it was loaded and RUN as a Basic program, or loaded as machine code program or started with SYS 49152, or was it 12768?

The menu program presented here will make life much easier. This program will produce a menu on your disk which when loaded and RUN as a Basic file will offer you a menu of the programs on the disk. Pressing the letter next to the program that you require will cause the program to be loaded into the computers memory and then executed as required.

Using the program

When RUN the MENU program will read the filenames off the disk that it is in the drive when requested. The user can then select which programmes he/she wants to appear in the menu. If you don't want a certain file in the menu just press 'N' when prompted. If you require a file to be present in the menu then pressing 'Y' will give you further options, asking for the type of file, etc.

The file type can either be Basic, press 'B' when prompted, or machine code 'M'. If you select Basic then the menu generator will move onto the next program on the disk. Selecting a

file marked with a 'B' will cause the program to be loaded and RUN just as you would with a normal program.

Should you press 'M' when prompted for the file type you will then be asked for the start address of the machine code program. You can give the start address either in decimal (e.g. 49152), or hexadecimal by prefixing the number with a dollar sign (e.g. \$C080).

When you have been through all of the programs on the disk the menu generator will save a program called "MENU" onto the disk. Loading and running this file will produce a menu on screen which you can load the required file from simply by pressing the relevant letter.

If you have a directory designer it is quite useful to name the program MENU so that it is the first in the directory. This means that you can load it into your computer with a simple LOAD "*" command.

Other options

As well as allowing you to create a MENU program the menu generator also allows you to specify a colour for the word MENU when it appears on the screen. The option to add a line of descriptive text to the menu also exists. Should you ever require to check that the disk in the drive is the one that you want to add a menu to, the main menu of the generator program offers the

facility of printing a directory listing to the screen.

Other options

As well as allowing you to create a MENU program the menu generator also allows you to specify a colour for the word MENU when it appears on the screen. The option to add a line of descriptive text to the menu also exists. Should you ever require to check that the disk in the drive is the one that you want to add a menu to, the main menu of the generator program offers the facility of printing a directory listing to the screen.

GETTING IT IN

HOW BASIC

- 1) Type in the BASIC program presented here.
- 2) SAVE the program with disk.
- 3) Turn OFF.
- 4) Enter the following:

```
POKE 40,0:POKE 46,10:RUN
```

- 5) LOAD and RUN the program saved in 2.
- 6) When finished enter the following to SAVE the program:

```
POKE42,1:POKE44,8:POKE46,100:  
POKE48,17:SAVE"MENU"OK
```

- 7) The program will now be on disk.

C64 UTILITY

PROGRAM: MENU MAKE.BAS	58	208 data 169,147,133,44,169,1	146,1462
		4,133,45,169,11,140,124,1,167	518 845,8 169,13,31,210,275,31
		33,33,33,33	1384,125,149,8,32,190,53,169
59	1 rem *****	248 data 138,20,32,238,288,281	133,132,2025
		48,138,249,261,32,174,248,28	24 521,32,158,238,149,8,32,169
		1,48,248,288	1395,148,15,142,8,169,19,32,1
60	2 rem * program to set up	60 254 data 93,261,50,240,3,76,1	86,295,2697
		13,18,78,256,18,268,8,141,91,	64 538 845,8 13,182,265,142,15,32
		56,1488	1394,253,264,144,288,9,32,287
		10 268 data 32,87,17,248,0,169,9	130,32,2293
		2,18,248,8,32,218,238,268,268	60 548 845,8 18,220,78,249,17,32
		248,1695	1384,229,32,218,268,148,148,9
		60 270 845,8 32,238,238,232,0,240	2,218,238,2697
		-249,76,-4,18,8,32,32,32,32,	60 558 data 168,9,177,299,261,24
		-1441	248,3,76,105,18,189,18,133,3
61	0 rem * save commands before	61 288 data 32,30,52,32,68,82,82	11,189,1844
		169,83,32,65,32,75,68,89,8,88,9	58 588 data 43,32,218,268,52,264
		0	1308,32,238,265,266,7,148,18,1
62	0 rem * close program	62 288 845,8 169,2,142,16,16,32	268,268,2618
		2,17,169,1,32,36,18,168,8,32,	68 638 data 248,19,268,78,268,24
		1644	1,169,9,32,38,76,102,28,17
		64 304 data 267,275,261,13,248,8	4,98,1738
		-173,228,28,288,182,36,144,24	68 658 data 32,264,74,15,169,4
		1,169,2,2384	132,38,28,32,268,255,281,77,2
65	20 For i=0 to 63:print i:for j=0 to	71 318 data 32,38,20,32,182,18,1	60 668 data 52,201,84,288,245,18
		32,38,18,18,141,152,65,169,8,18	9,2,32,38,25,168,8,141,145,19,
		2,244,1644	143,1630
66	20 268 845,8 168,16,32,189,275,18	66 328 845,8 168,16,32,189,275,18	68 688 data 148,18,52,34,169,78,1
		9,8,168,8,168,1,32,184,268,32	1,16,142,8,32,178,275,18,17
		1,392,1897	75,1640
67	40 268 data 268,142,8,52,208,275	67 338 data 268,142,8,52,208,275	68 698 data 250,268,8,32,38,
		1449,82,161,160,169,21,119,32	18 32,32,138,30,14,16,32,169,18
		3,168,18,141,152,65,169,8,18	18,18,18,18,18,18,18,18,18,18,18,
		2,244,1644	143,1630
68	648 845,8 32,218,255,169,3,32,	68 348 845,8 32,218,255,169,3,32,	64 628 data 78,105,28,142,0,32,2
		268,268,2680	67,265,261,13,268,8,132,134,2
		60 368 data 250,268,2,138,231,18	9,268,1891
		8,251,137,45,144,237,208,8,18	54 638 data 182,8,144,24,173,12
		5,200,137,26884	4,19,201,36,248,65,169,18,141,
69	268 data 64,164,268,169,265,3	69 388 data 64,164,268,169,265,3	143,18,1888
		132,265,169,264,265,169,8,32	18 648 data 148,148,18,170,138,2
		149,265,2688	8,104,18,72,281,48,144,28,19
70	678 845,8 78,4,14,64,58,77,69,	70 428 845,8 78,4,14,64,58,77,69,	9,144,19,1397
		78,82,148,8,168,13,32,259,265	61 658 data 24,138,134,18,141,14
		1,1684	4,19,273,248,18,228,138,19,14
		60 488 data 92,218,255,169,32,32	1,145,19,2620
		1,119,215,2678	64 668 data 274,84,238,1,76,248,
		390 845,8 255,169,64,32,148,32	18 688 data 182,138,138,65,219,76,
		0,32,307,278,281,13,248,8,182	11,19,170
		1,17,13,1444	68 678 data 238,169,8,242,244,18
71	488 data 180,182,31,144,241,1	71 488 data 180,182,31,144,241,1	1,142,145,28,142,8,189,134,18,
		52,164,57,168,17,32,189,261,1	281,58,1644
		89,13,32,2647	71 688 data 244,2,54,232,7,72,28
		412 845,8 238,238,234,234,234,234	18 698 data 268,173,148,18,24,138
		48,18,234,234,234,234,234,234,	68 888 data 38,144,144,18,175,14
		4,234,234,2768	8,18,122,149,18,141,145,18,18
72	428 data 234,234,234,234,234,	72 428 data 234,234,234,234,234,	4,54,232,2680
		234,234,234,234,234,234,234,234,	70 928 data 2,78,238,18,138,248,
		54,234,234,234,2694	232,232,142,1,276,267,64,160,
		438 data 234,234,234,234,234,	8,146,2880
		234,234,234,234,234,234,234,234,	68 918 data 247,18,177,289,268,2
		48,18,134,134,134,	18,148,1827,268,248,19,173,
		68 448 data 254,147,32,218,268,18	148,18,277,2645
		89,8,133,144,169,1,142,268,18	68 928 data 269,261,94,240,21,26
		6,18,32,1888	1,32,179,1,24,265,64,272,147,
73	488 data 180,255,169,8,168,8,	73 488 data 180,255,169,8,168,8,	70,145,2768
		948,8,32,188,255,32,182,255,18	68 938 data 44,238,148,18,238,14
		82,8,2672	8,179,78,261,18,268,6,178,147,8
74	68 428 data 32,268,268,248,142,	74 428 data 32,268,268,248,142,	8,144,1827
		68,73,32,267,266,32,267,265,32	74 948 data 44,238,147,18,238,14
		3,187,265,2658	7,18,238,147,18,268,173,144,8
		478 data 32,267,268,32,267,262	8,144,4,1888
		2,184,148,268,24,132,261,32,2	70 958 data 268,173,148,18,142,4
		87,222,132,2678	4,245,44,238,148,148,18,133,48
75	488 data 252,32,147,18,168,32	75 488 data 252,32,147,18,168,32	18,238,1825,50,84,149,29,133
		4,238,268,132,267,268,144,14	211,52,78,67,149,32,32,210,
		4,238,30,2612	875,32,7810
76	488 data 170,240,8,32,220,275	76 488 data 170,240,8,32,220,275	80 778 data 220,228,32,268,268,2
		1,78,170,17,189,13,12,218,205,	2,218,268,74,238,268,24,234,
		165,187,2217	234,234,234,1378
77	988 data 361,42,248,12,173,81	77 988 data 361,42,248,12,173,81	67 788 data 274,234,234,234,
		18,248,9,32,12,18,168,4,268,	

Disk Command Summary

To send a command to the disk drive use :

OPEN 1,8,15,"command";CLOSE 15

LOAD

LOAD "file",s	Load to start of Basic.
LOAD "file",s,i	LOAD file to address given in use count from.
SLoad "file"	LOAD Basic file in Basic 7.0.
RLoad "file",Block,Format address	Load file to different address. Basic 7.0 only.
BOOT "file"	Load and execute file Basic 7.1

SAVE

SAVE "file",s	Save a Basic file.
SSAVE "file"	Save a Basic file in Basic 7.0.
RSAVE "file",Block,Pa To Pa	Save code in Basic 7.0 where: s = Start address = End address.

FILE ACCESS

OPENfile-no,s,mode-no,"file-name,file-type,direction"	Open disk file where: File-type = P,U,B etc. Direction = R for read or W for write.
COPEfile-no,"file-name", Start-no,End-no	Open disk file Basic 7.0 R, W only needed for write.
CLOSEfile-no	Close open file
COLDfile-no	Close file Basic 7.0
PRINTfile-no,data	Send data to file.
GETfile-no,variable	Get data from file.
DEPUTfile-no,variable	Input data from file.

DIRECT ACCESS

"B";Device-no,sector-no	Seek track/sector as used.
"B";Device-no,sector-no	Write track/sector as used.
"B";Device-no,Block-no device-no	Execute code at track/sector.
"B";channel-no,byte	Save to byte in disk buffer.
"R";channel-no,Device-no, device-no	Read track/sector into buffer.
"W";channel-no,Device-no, device-no	Write buffer to track/sector.
"R";Device-address-(ICBB address)-ICBBfile of track	Send disk memory at address.
"W";Device-address-(ICBB address)-(ICBB) of track ICBBfile(ICBB)device.	Write to disk memory at address.
"B";Device-address-(ICBB address)	Execute sector code in ICBB at address.

TRYING TO USE YOUR COMPUTER?...

YOUR

COMMODORE

CAN HELP.

**GREAT
NEW RATES**



12 issues UK £13.50
12 issues Middle East
£21.50
12 issues Far East
£24.00
12 issues Rest of
World £21.40

Send your remittance to:
**INFOSYS LTD., 5 River Park Estate,
Ardubarnet, Mares, MP4 1AL.**



Please sign my subscription to this magazine with the
following details (or send me a separate order form):

Name Title

Address

Postcode

Signature

Date

Price per issue

Subscription period

Subscription total

Subscription start date

Subscription end date

Subscription type

Subscription status

Subscription reference

Subscription comments

Phone 01235 333333
Fax 01235 333333

BE IN ON THE ACTION . . . FROM THE VERY START



COMMODORE DISK USER is a lot more than just another computer magazine. Every issue carries a diskette containing more than £20 worth of software ranging from serious programming utilities to arcade games. There are plenty of Commodore magazines on the market, but we believe that this is the first to cater for disk users of all ages and tastes.

The **MEDIAWAVE** first issue will include on disc:

- ★ **3 into 1 Plus** - A superb all-purpose graphics editor

- ★ **Blenders** - An extremely addictive arcade strategy game

- ★ **Disk Designer** - Revamp your disk director, any way you like it.

- ★ **Ski Run** - A competitive 3D winter sports simulation. Break a leg!

... and lots more.

The magazine will include full details of how to use the programs, plus a wealth of reviews and features. This month we interview the US software houses poised to invade the UK. Computer tells us all its secrets, and you get the chance to win a selection of Microprose software. Plus news, reviews, and our mind-bending puzzle page.

COMMODORE DISK USER is what you have been waiting for - take out a subscription TODAY!

SUBSCRIPTION RATES

£15.00 for 6 issues U.K.
£18.00 for 6 issues EUROPE
£18.20 for 6 issues MIDDLE EAST
£19.30 for 6 issues FAR EAST
£18.40 for 6 issues REST OF WORLD
Airmail Subscription Rates on Request

MF

Send your remittance to:
**INFONET LTD., 5 River Park Estate,
Berkhamsted, Herts. HP4 1HL.**

Send your subscription to **COMMODORE DISK USER** with the
Invoice, or direct payment via **1**
or Visa 1. **Invoice Acknowledgement No.** _____
Card No. _____
Mag. No. (if applicable) _____
Address _____
Postcode _____
Signature _____
Date _____
I agree to pay my subscription to **COMMODORE DISK USER** for _____ issues.
I agree to pay my subscription to **COMMODORE DISK USER** for _____ issues.