

Computer manufacturers have a problem when deciding which sort of operating system to provide with their products. Unless the company is a giant, with giant resources, like IBM, it finds it very difficult to get applications software writers to write packages for a new machine. It is, if you like, a Catch 22 situation. Software writers are loath to devote time and money to a computer that has yet to show large sales. If they write packages and the computer fails to sell in large enough numbers, they obviously end up out of pocket.

Unfortunately, a computer without any applications packages is not much good to the user and won't sell. In microcomputer parlance this situation could be termed an 'endless loop'.

So the only solution for many small and medium-sized manufacturers is to design their computers to use one of the industry standard operating systems. But an 'industry standard' is created by its being around long enough to accumulate packages written to run under it. So the tendency has been for potentially innovative hardware to be compromised by the need to accommodate an operating system designed for an earlier age.

This scenario particularly applied to CP/M. But it is possible to compromise. Torch Computers was a typical case of a new manufacturer faced with the situation mentioned above. The Torch solution was to develop a CP/M-compatible operating system, the idea being that it would be an improvement on CP/M but still provide Torch users with access to the huge library of CP/M software.

Torch claims its version of CP/M, called CPN, is friendlier — easier to use and understand. As well as being the standard operating system for the Torch, it is available as an upgrade option for the popular BBC. Torch is offering a Z80 processor with 64K RAM on a

CPN BY TORCH

card which fits inside the BBC, plus twin 5.25in drives to go under it, effectively turning the BBC into an economy version of the Torch itself. (Full Pro-Test of the Torch Pack, PCN issue 16).

CPN actually stands for Control Program Nucleus. Unlike CP/M, which is loaded into the system from disk every time the computer is switched on, Torch has stored the CPN operating system in ROM in the machine.

Torch claims the built-in CPN commands are essentially the same as those found in CP/M. But as you would expect, there are a few differences.

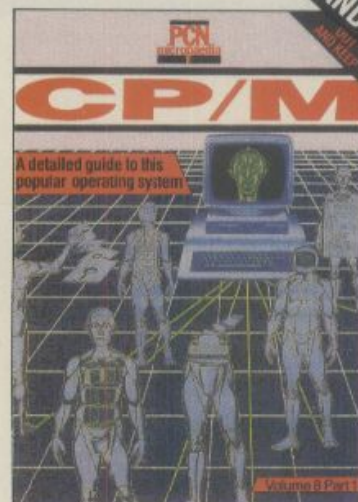
The table below compares some of the commands.

Under CPN you can't just go out and buy or borrow a CP/M program and expect to be able to run it — there is always the thorny problem of disk format compatibility. For most users the range of packages available from Torch Computers should fill most needs. According to Torch there are around 200 packages available from dealers for the Torch and Torch Pack. These include the popular WordStar, Supercalc, and Dbase II.

As an extra bonus Torch is offering a free 'bundle' of applications software with the Torch Pack comprised of Perfectwriter, PerfectCalc, PerfectFier and Perfect-Speller. Torch and CPN seem to have overcome the endless loop syndrome and established CPN among dealers.

Torch says that the rationale behind the release of the Torch Pack was to play the numbers game to the advantage of CPN. The more systems being used, the more dealers are prepared to supply applications software to run under CPN. Sheer volume of software being supplied also has a lowering effect on prices.

As an alternative form of CP/M, CPN must make an attractive offering to users who want a wide range of software options.



VOLUME 8

The table below compares CP/M commands with Torch CPN commands. CPN stands for Control Program Nucleus and allows the user to run CP/M programs on the Torch, although there are snags.

CP/M Command

DIR
ERA
ERAQ (MP/M)
MOVCPM
PIP

REN
SAVE
STAT dev:=dev:
STAT DEV:
STAT filename
STAT filename \$ r/o etc.
STAT d:
STAT USR:
SUBMIT
SYSGEN
TOD (MPM)
TYPE
USER
XSUB
FDISC

Torch CPN Command

DIR CPN Command can be given several filenames and also options.
DELETE Prompts for each filename (like ERAQ). The [u] option stops prompting.
Not Required.
COPY, DUP, INPUT, or PRINT depending on operation desired.
RENAME Different syntax. Allows multiple renaming. Identical to CP/M Version 2.2.
No Equivalents at Present.
Some facilities provided by *FX2 and 3.
[L] and [P] Options of PROTECT.
PROTECT.
[S] option of DIR.
Similar To USER Command.
COMMAND (C) Works slightly differently.
Not Required.
TIME.
Same.
Same as CP/M Version 2.2.
Provided automatically by COMMAND.
FORMAT and DUP.





CP/M: SOFTWARE

CODES FOR CONTROL

Control codes are one of the easiest methods of communicating with your micro under CP/M, so to help you with that communication, here is a list of those codes and what they do:

Control C — Restarts CP/M or aborts to the command processor if you're running a program.

Control E — Moves the cursor to the beginning of the next line. Is often used when you want to type in command lines without hitting Return.

Control H — Deletes one character to the left of the cursor.

Control I — Often referred to as TAB — its function is to move the cursor forward eight columns and it can be used in text files.

Control M — Same as Return.

Control P — Turns on the list device. This is usually the printer, but that can be changed using PIP. It's a toggle switch, so to turn it off you have to enter Control P again.

Control R — Displays the previous line as it was left in the buffer.

Control S — Stops the display but doesn't stop processing if a program is running.

Control U or Control X — Cancels the current line and stops it from being sent to CP/M.

Control Z — marks the end of a file.

If you have decided to get a CP/M based system, you'll have to ask some questions about the system. What does CP/M work with, what will it work with, how can it make life easier for you and what about expanding to a better system?

There are answers to all these questions, but the solutions themselves may not be that easy to implement. We all know that software running on one CP/M machine should run on another, but — and this is the important question — how do you get the software on your system?

The point is that although CP/M is a 'standard' operating system, there are sometimes slight differences in the way it works on different systems. This is important because transportability is the name of the game.

Micros running CP/M tend to be either 8080 or Z80 based machines. There are three ways in which to get a CP/M-based micro. First there is the manufactured machine, which has an 8080 or Z80 processor as standard. Many of these used to be S100 machines to which you added more memory by slotting in another board.

But the technology has now moved on, and it is possible to obtain a Z80-based machine with 64K memory, disk controller, I/O facilities, keyboard and screen all in one box. These aren't S100, because the memory and processor are now on one board instead of several.

The second type of CP/M machine is the upgrade one. These are micros, like the Pet and Apple, which have expansion capabilities. These two machines have the 6502 microprocessor, and therefore do not qualify as CP/M machines, but various Z80 boards which plug into/onto the Pet or Apple are available. These extra boards are normally Z80 processor boards with some memory (up to 64K) which allow 6502 machines to run CP/M.

The third type of CP/M machine is really constructed

by the hobbyist. Here, the hobbyist first designs the specifications for a CP/M machine, perhaps to consist of a Z80A processor, plus 64K RAM on one board, with I/O, disk controller and RS232. The need for expansion should also be taken into account, so some sort of bus system is often included.

To get the system together, you obtain the different parts from various manufacturers and put them together like a Lego kit, then put a fancy box around it.

With each type of CP/M machine there is one more step to go through before CP/M can be used. The manufacturer of the micro or add-on boards needs to get a licensing agreement from Digital Research to run CP/M on the kit.

There are now hundreds of CP/M machines, but the systems future is more debatable. Some say that the percentage is decreasing due to the current popularity of 16-bit machines. However, CP/M86 operates on 16-bit machines, and is basically (in operation) the same as 8-bit CP/M. And there are too many 8-bit CP/M systems and users around for anyone to try to sweep them under the carpet. Software houses will still be producing CP/M software for some time to come.

The problems of getting CP/M programs written for one machine to run on another are numerous. First, if you wish your programs to be transportable, then they shouldn't include fancy machine-dependent features. If you cannot avoid this, then have the machine-dependent features as a separate module from the main body of the program.

The most common method of configuring a program to run on several machines is to set up a Config file. First there is the main body of the program which is held as one or several files, then you have a small sub-program which can either pass to the main program the parameters for screen handling or store the parameters, which are 'collected' by the main program.



WARE & SYSTEMS

When run, the Config program should ask the user what keys are going to be used as cursor keys, what is the control code for inverse video, what is the code for clear screen and home cursor and so on. When the answers have been collected and stored in the Config program, all is well. When the main program is run, it can read the Config program and use the data in it to do these jobs.

There is no 'standard' for 5¼in drives, and this poses a problem for anyone who wishes to transfer programs to another system. For the owners of 8in systems, things are a little better as the IBM 8in standard format applies, but remember, not all manufacturers follow this. But if you're unfortunate enough to have a system where you cannot simply transfer programs/data by using the same disk, then all is not lost. There are a couple of programs on the market which enable you to transfer programs to another system by using the RS232 port and/or printer port.

Two of these are R-COM4 and T-COM4 or BSTAM.

Both will allow you to transfer files from one machine to another, even over the telephone. Checking is done, and if any data is scrambled then the corrupted block is reread. The advantage to these programs is that they will allow you to transfer programs between different disk formats.

The disadvantage as far as R-COM4 and T-COM4 are concerned is that the sending as well as the receiving systems must have a copy of the transfer programs.

CP/M machines: top left, the Osborne Executive, above, the RML 380Z and below, the RML 480. The range of machines running CP/M is wide because there are only three criteria. Your micro must have (or be able to handle) a Z80-type processor; it must have disks; and it must have a reasonable amount of memory (64K recommended). If you have that, there's nothing to stop you — provided that somebody has an implementation of CP/M for your micro.



FACTS ON FILENAMES

One of the big snags about CP/M is that it only allows you to specify filenames with up to eight characters. This may seem a lot, but it is all too easy to accidentally overwrite a file because you used the same filename. It is best to organise your files into different types, and CP/M does this to an extent by creating backup files with the extension .BAK, or temporary files with the extension .\$\$\$.

You can organise your files much more effectively if you use the 'standard' file extensions:

.ASM Source file of an assembly language program.

.BAK Backup copy of a file, ED uses this.

.BAS Source file of a Basic program.

.COB Source file of a Cobol program.

.COM File containing an executable program or a transient command such as STAT.COM

.DAT Used to denote that the file contains data.

.DOC File of messages or documentation.

.FOR Source file for a Fortran program.

.HEX A hexadecimal file created by using the assembler ASM.

.INT CBasic executable program created from a .BAS file.

.LIB Library of source routines for programs.

.LST File containing a printable listing.

.MAC Source of an assembly language subroutine.

.MSG Similar to .DOC.

.PAS Source file for a Pascal program.

.PRN Printable file created by a text formatter.

.REL Relocatable binary file.

.SAV Systems file.

.SRC Source file from CP/M users group.

.SUB Source file of commands for the SUBMIT program.

.SYM Symbol table generated by a MAC program.

.TXT Used to denote that the file contains text.

You can, to an extent, get a form of batch processing CP/M by using the transient program SUBMIT. Here a list of commands is created in a text file (using ED) and given the extension of .SUB. So, to delete all .BAK files do a STAT and a DIR; enter into the .SUB file:

ERA A: *.BAK
STAT A:
DIR

CP/M FROM THE INSIDE

CP/M's flexibility, ease of use and software support make it attractive to the average user, but its structure means that it is also popular with system programmers. CP/M's internal structure allows you to write a program on one system and run it on another.

When you invoke CP/M, it is loaded into RAM and occupies two main areas. The first consists of 100 Hex locations (0000 to 00FF). The second is approximately 8K in length, contains CP/M itself, and resides at the upper part of memory.

Diagram 1 shows the memory map of CP/M. The first (or lower) section of memory is used as a systems communication area. Information can be sent here or retrieved from here. The upper part of memory where CP/M resides is divided into three more sections called the CCP (Console Command Processor), the BDOS (Basic Disk Operating System) and the BIOS (Basic Input Output Section).

Between the two areas is the TPA (Transient Program Area), used by the user program and data. Note that any CP/M .COM facilities, such as STAT.COM, use this part of memory as well, thus they are called transient functions.

Functions such as DIR are part of the CCP, and therefore do not take up any user memory. You can see from the memory map that the CCP does not have a fixed address. This is because CP/M is designed to allow for memory expansion.

The address of the CCP is found by looking at the contents of locations 6 and 7, although you don't have to do this as CP/M does it for you. The CCP itself occupies a fixed amount of memory and is 2816 bytes in length.

CP/M itself is made up of a number of programs. These do several jobs, from printing characters to the display device to reading characters from the keyboard. System calls are invoked whenever control is passed to CP/M. Control is passed back to the program when the desired function has been completed.

Each function is called from the systems communication area at the base of memory by placing the desired function number into the C register. A call is then made to function 5 in the systems area, which contains a jump to the BDOS and the necessary function is executed. The following illustrates this:

```
LD E, 'H' put character in register E
LD C, 2 put desired function number in C
CALL 5 CALL CP/M
```

When this code is run a character will be printed to the output console.

Notice how flexible this arrangement is. Take for example a programming language with this arrangement of calling fixed functions. Here if in the language we said PRINT "H", the interpreter would simply call function 2, which would print the character.

The writers of the interpreter wouldn't have to worry

about what the output was (printer, VDU, so on), where it is located in memory and what ports it uses. Instead a simple system call is made and CP/M's BDOS or BIOS does the rest. This is why it is possible to write programs to run on different CP/M systems.

As mentioned earlier, function calls are achieved by making a system call to the CP/M entry point, which is in the reserved area below 100 Hex. The entry point is the 5th location, and it is a jump instruction to the CP/M entry point which is the base of the CCP.

This is because memory may change size when extra RAM is added to the system. CP/M ensures that the address at location 5 is always updated with the correct address.

This is achieved by using a transient program called MOVCPM. Part of MOVCPM changes the address whenever a new memory image is being created for a different size memory. Here we can see that no matter what the memory size, the CP/M entry point will always be location 5, which will be updated whenever RAM is added by using MOVCPM.

As you can see from the example program, the first piece of information that is passed to CP/M is the call number of the program which you would like CP/M to execute. The call number is placed into the C register prior to the call to CP/M. The call number is read and the necessary program or routine executed.

But it will be necessary to pass information to the system as well as the call number. In our example you can see that we have to pass the character to be printed to the console device ('H'). To pass parameters (as they are known), eight bit values such as characters and small numbers are placed in the E register. For 16 bit values such as addresses the DE register pair is used. Values returned from CP/M are placed in either the accumulator (A), or the register pair HL (High Low).

There are several different types of system calls. The first type are the basic input/output system calls (to the BIOS), while the second are the disk input/output system calls (to the BDOS).

The BIOS deals with all input and output of the four logical devices which are supported by CP/M. The devices are as follows:

CON Console display and keyboard for both input and output. Note: this is the only logical device supported by CP/M which is an input and output device.

RDR: Reader (paper tape) for data input only.

PUN: Punch for data output only.

LST: List device (usually a printer) for output only.

The second type (the BDOS), deals with input/output to and from the disk file structure.

As well as the I/O functions described, CP/M supports the input and output of character strings. This is achieved through the console device (CON:). This is supported because it's easier to make one system call for a whole line of characters than a call for each character. The string type call is referred to as



buffered I/O, and to get the string of characters displayed on the console device a buffer must be set up in memory.

CP/M detects the end of the buffer by the '\$' sign terminating it. The address of the string is placed in the DE register pair and the call number is placed in the C register. The structure of the print buffer looks like this:

c1 c2 c3 c4 c5 c6 . . . cK \$

K is the number of characters and the '\$' signifies the end of buffer marker. The input buffer is of a similar structure, although there is a difference. It looks like this:

m k c1 c2 c3 c4 . . . cK

M is the maximum amount of numbers allowed in the buffer, K is the number of characters actually in the buffer. The buffer is filled until a carriage return has been detected or until the maximum number of characters has been reached.

If you are going to use CP/M for machine code programming, you may not want to bother with buffering when it comes to accessing the console. CP/M allows you to access the console directly without the need to set up buffers. The two system calls which allow you to do this are 11 (get console status) and 6 (direct console I/O). The first allows you to find out if the keyboard has been struck, and the second allows direct I/O to and from the console.

With every version of CP/M, some extra or modified function call seems to be added or amended. Version 2.0 of CP/M has some extra system calls which were not supported in version 1.4. One of them is call number 12, get version number which would be returned in register L. The number is held as a hexadecimal value and starts off with 20 Hex for version 2.0. So for 2.2 the number will be 22 Hex.

This call may not sound as if it has any use, but think again. If you have a program which will operate under version 2.2 of CP/M, then all that need be done is to interrogate function 12 to see if the value 22 Hex is returned. If not, then all is not well. The other function call is 0. All this does is a system reset (warm boot), and it can be called from within a program whenever you wish to pass control over to the CCP.

And there are a good number of reasons for bothering to find out more about the BDOS, as it operates with the essential peripheral in CP/M — the disk drives. What happens, for example, when you give the prompt A: to access Drive A in CP/M?

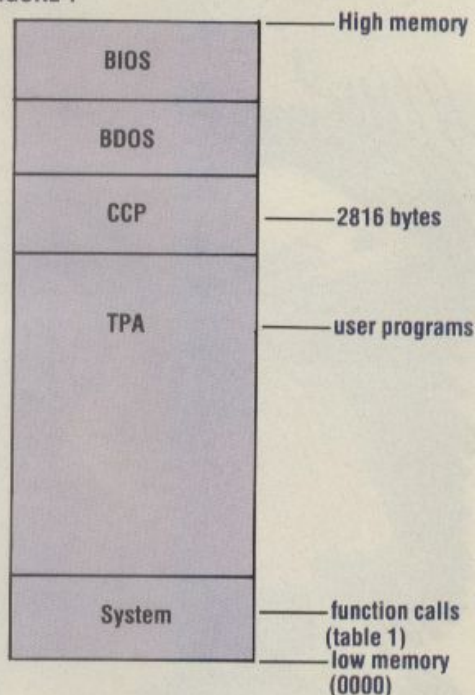
First, the Console Command Processor will interpret this and 'call' the necessary function in the system area (function 14). The system area would then communicate with the BDOS. Each file on the disk must be included in the DIR or directory, which sits on the centre track of the disk.

This is where the second part comes in, loading the directory into memory. The task is accomplished using a File Control Block containing the following information: the Drive Code, the File Name, the Extent, the Record count in current extent, the disk allocation map, information on the next record number to Read/Write and a Random Access area used for random access file maintained by CP/M.

Records can overlap — and may be split into two or three disk tracks, depending on their size. The BDOS of CP/M works out where the file and individual records are to be kept on disk.

For a full explanation of the BDOS, refer to the PCN Disk Drives Micropaedia Part II, page 92. For additional information on CP/M, contact the CP/M users group.

FIGURE 1



The memory allocation for CP/M begins at the low end with room for systems communication and an area for CP/M itself (not seen in this diagram). The TPA, or Transient Program Area, allows for programs and data — and it's followed by three additional areas: the CCP (of which we've already spoken), the BIOS and the BDOS. To boot CP/M, information is passed between BIOS and BDOS.

FUNCTION CALLS AVAILABLE IN CP/M (VERSION 2.2)

FUNCT	NAME	INPUT	OUTPUT
0	System reset	none	none
1	Console input	none	A=char
2	Console output	E=char	none
3	Reader input	none	A=char
4	Punch output	E=char	none
5	List output	E=char	none
6	Direct console I/O	Ref CP/M 2.2 guide	
7	Get I/O byte	none	A=0003
8	Set I/O byte	E=iobyte	none
9	Print string buffer	DE=string	none
10	Read console buffer	DE=buffer	string
11	Get console status	none	A=status
12	Get version number	none	HL=version
13	Reset disk system	none	none
14	Select disk	E=disk	none
15	Open file	DE=fcb	A=dir code
16	Close file	DE=fcb	A=dir code
17	Search for occurrence	DE=fcb	A=dir code
18	Search for next occurrence	none	A=dir code
19	Delete file	DE=fcb	A=dir code
20	Read sequential	DE=fcb	A=error code
21	Write sequential	DE=fcb	A=error code
22	Create new file	DE=fcb	A=dir code
23	Rename file	DE=fcb	A=dir code
24	Return login vector	none	HL=login vector
25	Return current disk	none	A=disk No
26	Set DMA address	DE=DMA	none
27	Get RBR address	none	HL=RBR
28	Write protect disk	none	Ref CP/M 2.2 guide
29	Get R/O vector	none	HL=R/O vector
30	Set file attributes	DE=fcb	Ref CP/M 2.2 guide
31	Get DPB address	none	HL=DPB
32	Set or get user code	Ref CP/M 2.2 user guide	
33	Read random	DE=fcb	A=error code
34	Write random	DE=fcb	A=error code
35	Compute file size	DE=fcb	Ref CP/M 2.2 guide
36	Set random record	DE=fcb	guide
37	Reset drive	DE=drive vector	A=0
38/39	No operation		
40	Write random record with zero	DE=fcb	A=error code

TAKE YOUR CHOICE

There may be a time when you decide to upgrade to a 16-bit system. Here you have a choice, and the two main contenders seem to be CP/M-86 and MSDOS. Each of the operating systems has its pros and cons, and it is up to you to weigh them up before making the plunge.

CP/M-86, like CP/M, keeps information about files on the disk, and the buffers of data are quite small. This means that read/writes tend to be slow. The speed of read/writes will vary greatly due to the fact that different manufacturers amend the BIOS for their own system. The worst feature (or lack of it) of CP/M-86 is the error recovery. The system will hang up if there is no disk in the drive and a read has been attempted. There is nothing to stop the manufacturer including some error trapping, although not many seem to be doing this.

On the plus side are the large help files available to CP/M-86 users. Also, the user guides are quite good.

If you wish to reconfigure the system, CP/M-86 allows you to do so quite easily with STAT.

The file management for MSDOS is quite fast, which means that disk input/output is very fast. The reason for this is because the information on file locations is held in memory rather than on the disk.

MSDOS shines as far as error recovery is concerned. Error recovery is part and parcel of the operating system, and the error messages on MSDOS are in a readable form.

Reconfiguring an MSDOS system can be a bit tricky. If you buy a new peripheral and want MSDOS to take notice of it, then some nifty fiddling with the operating system needs to be done.

You'll usually find some sort of configuration program with most MSDOS programs.

For example, in the CM/P Perfect Software suite of word-processing, spell checking, database and spreadsheet packages there is an extensive configuration program that you must run before doing anything else with the package. The program tests to make sure it correctly recognises all the attributes of the computer it's running on and then goes on to ask you about the peripherals you'll be running with the package.



THE BATTLE OF THE GIANTS

Computer technology has changed a great deal since CP/M first arrived, with major advances in CPU speed, RAM size, peripheral development and so on.

Luckily, CP/M is flexible enough to have coped with most of these developments. CPU speed changes or memory additions are no problem; a few boundaries are changed within CP/M, and the system will run.

The latest change is the appearance of 16-bit systems in large quantities. Alas, CP/M cannot be transferred to the new machines, so it has to be rewritten to run on the new systems.

One of the new manifestations is CP/M-86, written to run on Intel's 16-bit 8086 microprocessor. Digital Research, which developed CP/M, hasn't got the market to itself any more and there is competition to provide operating systems for the 16-bit family. One of them is Microsoft's MSDOS, considered by many to be the biggest threat to Digital Research's CP/M-86.

Fortunately for the user, many of the new 16-bit micros can run both MSDOS and CP/M-86. It is worth comparing the two systems to see what they do and how they do it.

CP/M-86

For those who have an 8-bit system and wish to expand to a 16-bit system, CP/M-86 offers the easier of the two means of upgrading. You are probably using 8-bit CP/M, and the commands on CP/M-86 are very similar.

You will not however, be able to run source code from your 8-bit system onto CP/M-86. To do this you will need a translator.

If you're lucky the source code (in Basic, for example) might run on CP/M-86 Basic, then you could transfer the source code to the 16-bit machine by using some kind of file transfer program. To do this you would need to set up a link between the two machines, probably through the RS232 port.

Once CP/M-86 is up and running, you will be confronted with the familiar 'A>' prompt. As you would guess, the default drive for CP/M-86 is A.

Inexperienced users would also be interested to know that you can invoke CP/M-86 on many systems by merely placing the systems disk in the machine and applying power. Some machines will ask you whether you wish to use MSDOS or CP/M-86, but the smarter ones will invoke one of the operating systems automatically.

CP/M-86 comes with a number of useful utilities, which include the text editor (ED), an 8086/8088 assembler (ASM-86), and a dynamic debugging tool (DDT). No doubt as time goes on, many other utilities will be released by the manufacturers and software houses.

Not all of these tools need cost you money — some are 'public domain' software and can be obtained from user groups or clubs.

The components of CP/M-86 are also similar to 8-bit CP/M. There is the Basic Input/Output system (BIOS),

the Basic Disk Operating System (BDOS), the Console Command Processor (CCP) and the Transient Program Area (TPA). The functions of these areas of CP/M-86 are as follows —

- BIOS — This area communicates with disk drives and other devices. This area is invisible to the user.
- BDOS — This controls the disk drives and also performs file maintenance.
- CCP — This reads the keyboard and processes commands. When you enter a command it is the CCP that interprets it and takes the necessary action.
- TPA — Your programs and data occupy this area, which can be changed to allow for memory expansion.

Of course, you will be looking for programs to run on CP/M-86 and there seem to be a few. Microsoft has not supported Digital Research's CP/M-86, even though many other manufacturers have released languages for CP/M-86. You can also get C, LISP and APL for CP/M-86.

MSDOS

This is the operating system from Microsoft. Although different in many ways, MSDOS and CP/M-86 perform the same function to act as an operating system.

Most of Microsoft's system languages are available to run on MSDOS. But it functions differently from 8-bit CP/M, so you'll have to learn a new set of commands if you are upgrading to it.

MSDOS is made up from three files: **COMMAND.COM**, **MSDOS.SYS** and **IO.SYS**. If you look at the directory of the systems disk, you will notice that both **MSDOS.SYS** and **IO.SYS** are not displayed, as they are hidden files.

There are four different disk areas with MSDOS. There are reserved areas (sectors) on the disk that contain information used each time MSDOS is loaded. The directory contains information on all files and will include information on the file specifier, size and time and date of last modification. The File Allocation Table (FAT) contains information on the data which makes up a file. Finally, there are the files themselves which don't necessarily have to occupy sequential disk space; they may be scattered.

The major difference between MSDOS and CP/M-86 is the way MSDOS treats files and devices. MSDOS supports device-independent I/O, and this means that the distinction between files and devices is internal.

The similarity between CP/M continues when we learn that MSDOS has internal and external commands. Here is a list of the internal commands available on MSDOS:—

- COPY** Copies specified files or file.
- DATE** Sets and displays date.
- DEL** Deletes specified files or file.
- DIR** Lists directory (may be specified).
- PAUSE** Pauses for input in a batch file.
- REN** Renames a file.
- REM** Displays a comment in a batch file.
- TIME** Sets and displays time.
- TYPE** Displays the contents of a specified file.

There are also a number of external commands with MS-DOS:—

- CHKDSK.COM** Checks specified disk and gives status information.
- FORMAT.COM** Prepares a new disk for storing data.
- DCOPY** Disk to disk copy program.
- RDCPM** CP/M-86 file transfer program.

PCDOS

Similar to MSDOS is PCDOS, the operating system

developed for the IBM PC. There are subtle differences between the two systems, however.

The first difference has to do with the IBM's interrupt-driven screen handling, and the second with the PC's system of program protection.

The interrupt problem is based around the fact that VDUs on most micros behave like the teletype machines that preceded them while cursor addressing is handled by ASCII characters.

Due to the differences in Basic under the two operating systems, you'll find yourself missing some MSDOS commands if you move to PCDOS — such as the **WIDTH** statement. Although extensions to the Basic are now becoming available, it may take some time before you can get them.

For more details on PCDOS and MSDOS conversions see *PCN Issue 6, Page 18*.

The systems compared

For the user, the choice of CP/M-86 or MSDOS may be irrelevant, but for the operator and/or programmer it can mean a lot. The operator needs an operating system that is easy to use, robust, flexible and understandable. The programmer needs a system with which he can manage files easily and program for error-trapping.

CP/M, CP/M-86 and MSDOS have separate functions which are called by the programming language or program directly. It's important for the programmer to know what the functions are. Here is a list of those functions:—

Future stock

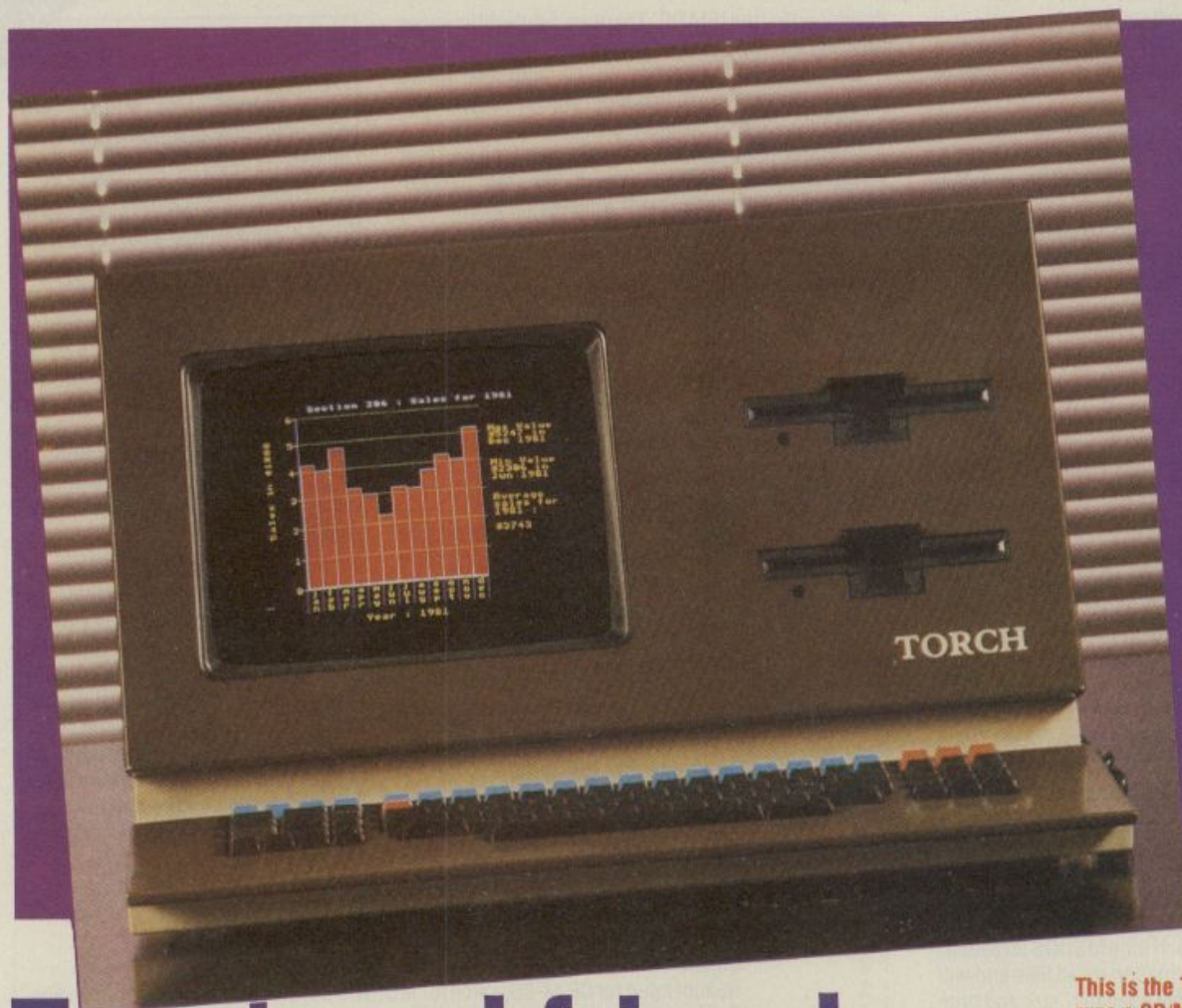
Both CP/M-86 and MSDOS are currently being updated. At the time of writing (July 1983) Concurrent CP/M-86 is still to be launched. This will enable several tasks to be done at the same time. You could, for example, be editing one letter while printing another.

It is also promised that future versions of CP/M-86 may run up to three times faster as a result of the transformation of the index keys by a method known as 'hashing'. There may also be some additional error trapping as CP/M-86 is somewhat lacking in this respect at the moment.

MSDOS version 2.0 is to have a new user interface with windows of information presented to the user. Online help files will also be available by entering '?'. Another important development will be that MSDOS will support a protocol for text and graphics. Networking an enhanced **DEBUG** and serial printer support is also on the horizon.

FUNCTION	DESCRIPTION
0	System reset for CP/M-86 and MS-DOS.
1	Console input with echo, for CP/M-86 and MS-DOS.
2	Console output; same as 1.
3	Reader input for CP/M-86, same for MS-DOS but only one auxiliary input device.
4	Punched output; same as 3.
5	List output; same as 3.
6	Direct console I/O for CP/M-86 and MS-DOS.
7	For CP/M-86 get IOBYTE . For MS-DOS direct console input with echo.
8	For CP/M-86 set IOBYTE . For MS-DOS console input with echo.
9	Print string for CP/M-86 and MS-DOS.
10	Read console buffer for CP/M-86, almost the same for MS-DOS.
11	Get console status for both systems.
12	Return version number for CP/M-86. For MS-DOS clear keyboard buffer.
13	Reset disk system. Same for CP/M-86 and MS-DOS.
14	Select disk. Same for both systems.
15	Open file. Same for CP/M-86 and MS-DOS.
16	Close file. Same for CP/M-86 and MS-DOS.
17	Search for first. Same for CP/M-86 and MS-DOS.
18	Search for next. Same comments for 17.
19	Delete file. Same for both systems.
20	Read sequential. Same for both systems.
21	Write sequential. Same for both systems.
22	Make file. Same for both systems.
23	Rename file. Same for both systems.
24	Return log in vector for CP/M-86, not used by MS-DOS.
25	Return current disk. Same for both systems.
26	Set DMA address. Same for both systems.
27	Get address of allocation vector. Same for both systems.
28	Write protect disk. Used by CP/M-86 but not MS-DOS.
29	Get read-only vector. Used by CP/M-86 but not MS-DOS.
30	Set file attributes. Used by CP/M-86 but not MS-DOS.
31	Get address of disk parameters. Used by CP/M-86 but not MS-DOS.
32	Set/get user codes. Used by CP/M-86 but not MS-DOS.
33	Read random. Used by both systems but note different failure codes for both.
34	Write random. Same as 33.
35	Compute file size in records. Same for both systems.
36	Set random record. Same comment as 35.

Note that although a given function may carry out the same task it will often carry out that task differently on different systems.



Front end friend

Although CP/M may seem a perfectly sensible operating system to people who've used it for a while, to beginners it's often baffling.

To remove much of that bafflement, software houses have designed 'front-end' programs which turn operating system commands into simple English. They often run as 'background programs' which must be loaded before running any CP/M programs.

For example, the Torch Mail Plus running on the Torch CPN CP/M lookalike system can be used as a front end for running all kinds of programs including word-processors, games and spreadsheets.

On booting up the machine, you immediately load the front-end program into the machine. A typical front-end will then offer you a series of menus, giving you single-key entry choices about program-loading, file-handling, disk formatting and device control.

On the Torch Mail Plus front end, you have a series of choices once the front end has been loaded. The first two choices let you use the machine's built-in modem to call another machine. The third choice lets you use the machine's internal clock to create an alarm call that beeps intermittently when it goes off.

The fourth choice is a summary that tells you what programs you've run and what calls you've made using Torch Mail.

This function is a good example of the background nature of front ends. The front end does all the house-keeping items like keeping track of the time, calling up programs and keeping a record of which programs have been called.

Torch's program also controls local file operations, deals with file and drive protection and keeps a list of telephone numbers and passwords.

To enter a program such as Wordstar from the front end Mail Plus program, you just choose Option 6 — Timed Command Execution and then type 'WS' for Word Star, and Mail Plus will load the Word Star program.

When you're finished you press the Word Star QUIT command. Mail Plus automatically reloads and presents you with the Mail Plus menu again.

In this case, the front-end menu also carries out the business of making phone calls, connecting to a networking system and taking care of the necessary access code and Obey files to carry out those operations out.

Other house-keeping tasks such as formatting, copying, printing, typing and renaming are also the responsibility of the front-end programs. In many front-end menus, you merely press an option such as '(a) Format the disk' or '(b) Backup the files in Drive A to Drive B' and the front-end program will give CP/M the appropriate instructions to carry out that task.

The theory behind the front-end is that many CP/M users, because they are often in business, don't necessarily have a great background in computing. To solve this difficulty, English language front-ends are seen as the solution.

So the average business user needn't know what COPY, FORMAT or PIPS mean in order to use those commands.

This is the Torch computer, which runs a CP/M lookalike operating system called CPN, as well as an extensive front-end program known as Torch Mail Plus.

Contributors: David Janda, Ian Scales and Geof Wheelwright
Design: Nigel Wingrove
Illustrators: David Fathers (Part 1) and Sophie Grillet (Part 2)

NEXT WEEK

We begin a three part series about the Commodore 64. The Vic-20's new big brother has recently taken a price plunge that will no doubt inspire the interests of many past, present and potential Commodore users.

We'll look inside the 64, how to improve on the machine's Basic, the vast array of serious business applications that can be considered for it, and the games it can play. We'll also give you a rundown on peripherals for the 64 and a discussion of its compatibility with the Vic.