Introduction to CP/M

John R Hudson

1 What is CP/M?[1]

CP/M (or Control Program and Monitor) is an operating system — not just any old operating system but the brainchild of Gary Kildall and the forerunner in concept of Linux. All computers need an operating system to make the hardware work together but, before Gary, every new type of computer had its own operating system.

A single operating system

Gary realised this was pointless because, not only did you have to write a new operating system for every computer, you also had to re-write all the software to suit the new operating system. Gary's solution was simple — he would write an operating system which looked the same to every piece of software, whatever the hardware and give the computer manufacturers enough instructions for them to finish the operating system off for their hardware.

Bill Gates

Gary's idea was so good that a teenager called William Gates III did all the necessary to enable CP/M to run on Apple II computers, thereby allowing Apple II computers to run all the sophisticated word processors, spreadsheets and databases which were being written for CP/M.

Gary built in facilities to add new hardware very easily and developed a way of dealing with different capacity floppy discs equally easily — something Bill Gates never really developed with his own MS-DOS.

AMSDOS

So, when Alan Sugar wanted to extend the appeal of his computers, CP/M was the obvious system to use. In fact, both AMSDOS and LocoScript on the PCW contain a cut-down version of CP/M to handle all the hardware.

But, because AMSDOS is such a good operating system and companies like Tasman, Arnor and Campbell Systems were happy to use AMSDOS, many users never bothered to look into CP/M and you only really need CP/M for specialist programs, for heavyweight programs like Newword, Supercalc or dBASE or for programs like LOGO and VDE which only run under CP/M.

[1]Originally an article submitted to WACCI on 5 July 1999

Loading CP/M

Because AMSDOS is already resident, you have to tell it to make way for CP/M by entering `|CPM` if you have CP/M on disc or `|EMS` if you have CP/M on ROM. The main practical difference between AMSDOS and CP/M is that AMSDOS automatically loads BASIC or, on the Plus machines, a game after it has loaded itself. CP/M doesn't unless you tell it to. In addition each operating system locates itself in slightly different parts of memory; so you cannot use the same programs with both of them.

Two flavours

There are several flavours of CP/M but the two most relevant to CPC users are CP/M 2.2 which sold over a quarter of a million copies in the early 1980s and CP/M 3.1 (or Plus) which sold several million on Amstrad machines alone. CP/M 3.1 was a complete re-write which, however, maintained backwards compatibility with programs written for CP/M 2.2.

The main practical difference is that CP/M 2.2 could only work with 64K of memory. At the time CP/M 2.2 was being developed you could get a Sinclair ZX80 computer with a 'massive' 1K of memory which could be extended by a further 8K. So 64K was generous but it soon proved inadequate mainly because of the development of graphics on personal computers. The CPC 664 has 16K of video RAM which meant that, once you took away that plus the space for CP/M 2.2, you only had around 43K for programs. Most other CP/M systems used text screens which need far less video RAM and you could get between 48K and 56K of memory for programs.

CP/M 3.1 got over this limitation by bank-switching — fooling the chip into believing that it was only dealing with 64K of memory when it was dealing with much more. This allowed the video RAM and most of the operating system to be stored in banked memory, leaving 61K for programs.

Space saving

In order to maximise the space available under CP/M 2.2, Gary Kildall had already arranged for certain parts of the operating system to be stored on disc and overwritten in memory when a program is running. Then, when the program finishes, CP/M 2.2 recalls these parts of itself from disc. This is why there are two full reserved tracks on a system disc and why you have to leave a system disc in drive A: under CP/M 2.2. You can remove it while certain programs are running but you have to put it back before you exit from the program.

Once CP/M 3.1 could use banked memory, this was no longer needed as any bits which are overwritten can be recalled from banked memory. But, for compatibility, all Amstrad CPC system discs still have two reserved tracks.

Instructions

Once loaded CP/M simply follows the instructions it is given; these are either the six built in instructions (`DIR`, `ERA`, `TYPE` and so on) or those in files on disc. Keeping most of the instructions as files on disc saved memory but it also gave CP/M a flexibility which no other micro-computer operating system until Linux has had.

There are three types of instructions in these files (which all end in .COM) — those which carry out additional operating system functions like formating discs and copying files, those which extend the facilities of the operating system like PUT and GET and those which start the things people are really interested in like word processors, spreadsheets, databases and accounts programs.

Some of the first group, like `DIR` and `TYPE`, have the same names as the built-in commands but are extended versions of them, allowing you to do more things.

The second group are like `|` comands in AMSDOS — they extend the operating system. Their disadvantage is that they can normally only do this by taking up additional space in memory. For example, most ROMs can only be addressed under AMSDOS through a `|` command and, because these take up memory, most have to be switched off before you can use Mini-Office II which takes up all the available memory.

For this reason, CP/M instructions like `PUT` and `GET` which behave like `|` commands are programmed to detach themselves immediately their work is done unless you instruct them to remain.

Creating the environment

Because CP/M is a generic operating system which presents the same face to every program, programmers need not know anything about the computer on which their programs will run. The manufacturer, in our case Locomotive Software acting for Amstrad, writes all the programs needed to get the best out of the hardware and supplies them with the machine.

In the case of the 6128 these are

`PALETTE.COM`	`LANGUAGE.COM`
`SETKEYS.COM`	`SETSIO.COM`
`SETLST.COM` and	`SET24X80.COM`

which control the screen colours, language, keyboard layout, serial interface, printer configuration and screen size respectively.

Because the CP/M 3.1 operating system is stored in banked memory, these programs can be used at any time to change any aspect of the way the hardware responds. So, for example, I have different keyboard definition files for each program I use. Running `SETKEYS` before I load a program ensures that the function keys operate as I want them to. When using Supercalc2 I normally have the function keys set up to imitate Supercalc5 on the PC but, if I want to use them as a numeric keypad, I put them all into supershift.

Macintosh, IBM and Microsoft have all put a lot of effort into developing common interfaces for their products so that the same actions produce the same results in every program. The CP/M approach is the exact opposite — you can have a different layout for every program you use — and we will help you to achieve it.

Things are not so easy under CP/M 2.2 — there wasn't the demand because keyboards were far less sophisticated — but Locomotive Software did supply some utilities to configure the 664. You cannot change the configuration on the fly as you can with CP/M 3.1 but you can keep a separate system disc for each configuration you want and load CP/M 2.2 afresh each time you want to change the configuration.

Automation

CP/M was developed when most users did not sit at a screen; they ‘submitted’ their jobs to the operator who ran them as part of a ‘batch’. So CP/M came with facilities to run a batch of jobs and these facilities were extended under CP/M 3.1 into a fully fledged scripting language which allows you to automate everything you do — see Chapter 3.

User input

Because the operating system is made up of a core plus files on disc holding further instructions, it is relatively easy for users to add their own instructions. A program like `CLS.COM` is in the same class as `PALETTE.COM`; it will only operate on certain hardware (specifically the Zenith/Heath terminal adopted by Amstrad for all its CP/M machines). A progam like `PAUSE.COM` will operate on any CP/M system while a program like `SC2DATE.COM` or `XDATE.COM` will only operate on CP/M 3.1 systems. There is little you can do if someone tries to use `PALETTE.COM` on the wrong hardware but it is quite easy to program a check that the operating system is CP/M 3 into `SC2DATE.COM` and `XDATE.COM`; so nine times out of ten, you will get a meaningful response from a CP/M instruction even if it is — sorry, I can’t do that!

The whole PD scene — of which WACCI is a part — arose because CP/M lends itself to user input. Except for the hardware specific programs like `CLS.COM`, you don’t have to know anything about the hardware on which your program will run; you simply have to know about CP/M. As with Linux, this ‘open architecture’ encouraged people to develop programs in a way which no other operating system before Linux has. In some cases, such as GSX, these programs were so good that Digital Research adopted them as their own and developed them. In others, like VDE, they offered features to the user which were ahead of commercial programs.

Though PCs had taken over the commercial world by the end of the 80s, it was only the arrival of desktop publishing and the Internet in the 1990s that finally gave the edge to MS-DOS over CP/M. By that time, all its best features had been included in DRDOS which is still a superior operating system to MS-DOS and available from Caldera as Open DOS.

2 All those funny keys

Anyone who uses VDE, Supercalc, dBASE or a number of other older programs may wonder why they use such funny keys, especially for moving the cursor around.

The reason is that arrow keys only became a regular feature of micro-computer keyboards after the arrival of the IBM PC which, in its earliest version, had them on the numeric keypad. The first micro-computer I used had a typewriter keyboard plus `ESC`, `CTL` and `LF` (line-feed). So software writers had to assign some keys to controlling the cursor and other necessary functions.

Printing

Everyone agreed that `CTL-P` should turn the printer on and off; so in CP/M and dBASE it does just that. If you want to print out a directory listing in CP/M, enter `DIR<CTL-P><RETURN>` and, as long as your printer is on, you will get a printout of whatever appears on screen. Press `CTL-P` when it has finished or you will get all your subsequent commands printed as well! In VDE, WordStar and all similar word-processors, `CTL-P` initiates a printer control sequence, the second letter specifying exactly what should happen but this is only acted upon once you have begun printing; so `CTL-PB` prints bold, `CTL-PD` is doublestrike, `CTL-PS` is underScore and so on. As with `CTL-P`, you repeat the command to stop the effect.

Cursor movement

There were several attempts to agree on which keys should control cursor movement and we are left with three on the CPC. The most common is the ESDX diamond in which `CTL-E` moves up, `CTL-X` moves down, `CTL-S` moves left and `CTL-D` moves right — an arrangement I had used for some six years before getting a CPC and some arrow keys to play with!

CP/M uses `CTL-A` to move left and `CTL-F` to move right and LOGO uses yet another group of keys for cursor movement but, if you follow the instructions for installing LOGO, you will find you can ignore these and use the arrow keys anyway. So we will pass over them swiftly.

The most likely reason for wanting to use `CTL-A` or `CTL-F` in CP/M is that you have spotted a mistake in a long command like

```
SET B:[ACCESS=ON,UNDATE=ON
```

and you want to go back and correct `UNDATE` to `UPDATE`. You can use `CTL-A` to get back to the N and then delete it either with `CTL-G` if the cursor is over the letter or with `CTL-H` if it is over the `D`.

Deleting

Many older programs use `CTL-G` and `CTL-H` as the equivalents of CLR and DEL but some, like Supercalc, follow the purist definition of BACKSPACE, which is what `CTL-H` replaced on the typewriter, and simply backspace the cursor without deleting! So you need to try them out on older CP/M programs to find out how they work as generic CP/M programs often do not know about and will not respond to the CPC keys.

After making such a correction in a CP/M command, you can simply press `<RETURN>` and the command will be carried out. You don't have to go to the end of the line as in a word-processor.

Carriage return and line feed

When you are using a word-processor `<RETURN>` normally sends two codes — carriage return (CR) and line feed (LF) — which can be sent separately with `CTL-M` and `CTL-J` though, for most purposes, `CTL-M` can be treated as the equivalent to `<RETURN>`. The one exception is when using PIP. You can use PIP to create short files with commands like:

```
PIP example.sub=con:<RETURN>
```

When you enter such a command, PIP will engage in some disc activity and then go silent with the cursor at the beginning of the next line. You can then type in a command such as:

```
language 3
```

but, when you get to the end of the line, you cannot use `<RETURN>` because PIP cannot translate `<RETURN>` into its two separate commands; you have to use `CTL-M` followed by `CTL-J` and you will see them in action as you enter them. You can then add another line, such as,

```
setkeys keys.vde<CTL-M><CTL-J>
```

and so on until you have finished creating the relevant SUBMIT file. To finish the whole operation, you must enter `CTL-Z` which is the CP/M end-of-file marker. At this point, the disc drive will whir into action as `PIP` saves the file to disc.

Transferring files

`CTL-Z` is also worth knowing if you are a Notepad user or if you want to transfer files from another non-CP/M computer as you can transfer files to the CPC using `PIP`. I set my CPC serial interface to 4800 with `SETSIO 4800` and have the Notepad set to the same speed using the printer menu. I enter

```
PIP filename.ext=aux:<RETURN>
```

on the CPC and, from the document menu — not the printer menu! — on the Notepad `Menu-T-S`.

At the end of the transmission, `PIP` needs to receive `CTL-Z`. This is the right arrow key six characters from the right on the top line of the Symbol-Menu screen.

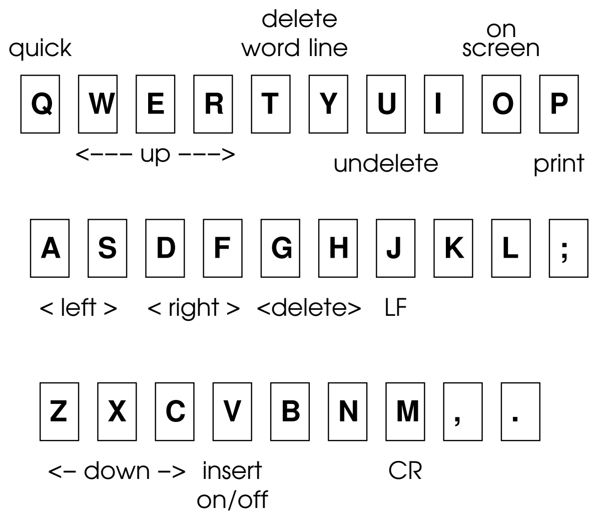

Figure 2.1: Common key definitions

You can either open a separate document with the single character ‘`CTL-Z`’ in it and send that after the file. Or you can add `CTL-Z` to the end of whatever file you want to send before you send it.

When you send `CTL-`Z, the disc drive whirs as `PIP` writes the document to file. As `PIP` holds everything in memory up to this time, you cannot use this technique for very long files but I have safely transferred files of over 30K and it is by far the simplest way of transferring files to the CPC using a serial interface.

An emerging pattern

We can now see a pattern emerging on the keyboard (Figure 2.1).

The cursor movement keys are all on the left of the keyboard while the delete and `<RETURN>` keys are together. This basic pattern then influenced the other combinations which people developed, though from here on, the differences exceed the similarities.

WordStar

On the left hand side of the keyboard, `CTL-R` and `CTL-C` were used in WordStar and dBASE to move up and down a screenful rather than a line at a time as with `CTL-E` and `CTL-X`. In WordStar and VDE, Q before a command meant ‘Quick’. So `CTL-QS` took you to the left of the screen, `CTL-QD` to the right, `CTL-QE` to the top and `CTL-QX` to the bottom. By extension, `CTL-QR` took you to the start of the document and `CTL-QC` to the end of the document.

On the right hand side of the keyboard, `CTL-I` was already being used for TAB and `CTL-L` for formfeeds by some programs so `CTL-O` and `CTL-K` were used by WordStar and VDE in similar ways to `CTL-P`. `CTL-O` changes the ‘On-screen’ appearance of the document while `CTL-K` initiates all the main editing commands.

`CTL-OC` centres text and `CTL-OL` and `CTL-OR` set the margins; `CTL-KB` marks the start of a Block and `CTL-KK` marks the end of a blocK; `CTL-KC` Copies, `CTL-KH` Hides, `CTL-KS` Saves a document, `CTL-KX` eXits and so on.

More deletion

In the middle of the keyboard, `CTL-Y` was used to delete a line and `CTL-T` to delete a word; `CTL-U` usually undeletes or cancels the last command — in dBASE it deletes or undeletes a record depending on its current status; so all the delete keys are together!

Finally, `CTL-V` is almost always used to turn INSERT ON or OFF.

When arrow keys and then function keys became commonplace on computer keyboards, Arnor and others were able to simplify this system. But though many people heaved a sigh of relief, it wasn't always the blessing it may have appeared.

Though there were differences between programs — save is `CTL-W` in dBASE, `CTL-KS` in VDE and `/S` in Supercalc — at least the keys were all in the same places on the keyboard. There is no agreement about where the arrow keys and the function keys on a keyboard should appear. So changing computer often means learning new locations whereas I can sit down to WordStar, VDE, Supercalc or dBASE on any computer and know that the combinations I learned on the very first computer I used fourteen years ago will be in the same places today. IBM has realised the advantages of the old system and is trying to promote a new system called Common User Access which will make the ways people have to interact with different software the same.

CP/M control keys

There are a few more useful commands from the early days which are worth knowing (Table 2.1). `CTL-W` or `CTL-R` recall the last line typed. `CTL-W` works with CP/M and is particularly useful when you have typed in a long command, pressed `<RETURN>` and got an error message because of a typing error. Using `CTL-W`, you recall the line and use `CTL-A`, `CTL-F` and `CTL-G` (or `H`) to correct the error.

You can also use it if you are searching for a file in a batch of high capacity floppy discs; if you have your CP/M system disc in drive A: and enter:

```
dir b:lostfile.ext[user=all<RETURN>
```

then every time CP/M throws up '`No file`', you can put in another disc, and press `<CTL-W><RETURN>` to get it to carry on the search on another disc without having to type in the whole command again.

Several programs, including CP/M's successor on the PC, DR DOS, use `CTL-R` in the same way though dBASE doesn't wait for you to press `<RETURN>`; it simply carries out the last command.

Finally, `CTL-C` is used in CP/M to cancel the operation of a command and can be very useful, for example, to stop a long document on which you have just used `TYPE` from scrolling all the way through.

Table 2.1: Useful CP/M control keys

A	move left
B	move to start or end of line
C	cancel
F	move right
G	delete letter under cursor
H	backspace (delete)
J	line feed
M	carriage return
P	echo/cancel echo to printer
W	recall last line typed
Z	end-of-file marker
	(there are others!)

3 CP/M Script[1]

If you have been keeping up-to-date with the PC and Mac worlds you will have noticed increasing emphasis on scripting. Apple has begun to develop AppleScript, Lotus, who brought out LotusScript several years ago, has advanced its cause in the most recent edition of SmartSuite and Microsoft has made its BASIC into a scripting language under Windows 95.

Ironically, all three have previously turned their backs on scripting languages, Apple and Microsoft because they thought a windows interface wouldn't need it and Lotus because they thought it would be sexier to have embedded macros rather than imitating Supercalc's scripting language.

But 664, 6128 and DD1 users can all use the grand-daddy of microcomputer scripting languages — CP/M — to do much the same things as Mac and Windows users are now being encouraged to do.

Entirely in scripts

While CP/M 2.2 can run scripts, CP/M Plus can run entirely on scripts. It is programmed to look for PROFILE.SUB as its first program to run. PROFILE.SUB usually contains a series of CP/M commands to customise your machine and load your favourite program. The .SUB ending tells CP/M to invoke `SUBMIT.COM` which places the commands in a special temporary file.

But the ability to use the commands in the special temporary file is present in both CP/M 2.2 and CP/M Plus. At the end of every program (including `SUBMIT.COM`) and before offering you the prompt, CP/M looks for the special temporary file; you may have noticed CP/M checking for this file when it appears to hesitate just before offering you the prompt.

If it finds the file, it reads, then deletes and executes the last command in the file. It does this each time it exits from a program until there is only one command left when it deletes the file. So if the last program leaves a new temporary file to execute, you will never return to the prompt.

Writing a menu

In WACCI 77 Uncle Clive showed us how to create a menu from BASIC for use with CP/M Plus. You can replicate Uncle Clive's example menu in CP/M Plus by creating two .SUB files each of which can be called from PROFILE.SUB. 1.SUB will consist of the line:

```
SC2
```

and 2.SUB will consist of the line:

[1]Originally submitted to WACCI in 1997

```
DIR
```

However, to get CP/M to respond to '1' or '2' on its own, you have to tell it to look for '.SUB' files rather than '.COM' files. The line:

```
SETDEF [order=(SUB,COM)]
```

in PROFILE.SUB will do this and you need to make sure `SETDEF.COM` is on the system disc.
PROFILE.SUB might then consist of:

```
SETDEF [order=(SUB,COM)]
;
;
; 1 Supercalc
;
; 2 Dir
;
; Enter 1 or 2 and press <RETURN>
;
```

As in assembler and some other languages, semi-colons are used to indicate the start of a comment. They can come after the command or, as in this case, at the start of a line so that the whole line is treated as a comment. If you don't like the semi-colons, you can get rid of them by creating a file called MENU with a wordprocessor.

```
1 Supercalc
2 Dir
Enter 1 or 2 and press <RETURN>
```

and amending PROFILE.SUB to read:

```
SETDEF [order=(SUB,COM)]
TYPE MENU
```

You can further refine this by adding the line

```
TYPE MENU
```

to the end of the 1.SUB and 2.SUB files so that, when CP/M has carried these commands out, it always returns to the menu, though adding it to 2.SUB may push a long directory display off the screen.
Because CP/M is showing you the prompt at the end of MENU every time anyone can break out of this sequence by typing an alternative CP/M command like PIP which does not result in a new .SUB file being created. So you may want to add a further line to MENU saying:

```
To return to this menu, enter
TYPE MENU and press <RETURN>
```

Submitting a batch

However, the idea for CP/M SCRIPT did not come because people wanted to write menus; the early CP/M computers had single line screens so could never have displayed a menu. The idea came from mainframe computers where you gave your jobs to a technician who, in order to use computer time efficiently, would 'submit' a 'batch' of jobs to the computer using a series of 'job control' commands.

You can use `SUBMIT.COM` to create the necessary temporary file in both versions of CP/M. But you need `XSUB.COM` to do some things in CP/M 2.2 which you can do automatically in CP/M Plus. This is the main downside of using CP/M SCRIPT; you have to have `SUBMIT.COM` on a disc to convert your commands in a .SUB file into a form CP/M can read and, if you want to do the more fancy things in CP/M 2.2, `XSUB.COM` as well.

Customising programs

Before looking at the 'job control' functions of CP/M, I will run through the main customising programs. To enable CP/M 2.2 users to customise the machine Locomotive Software provided `SETUP.COM` but CP/M Plus users are expected to set up the 'profile' of their machines every time they switch on by using PROFILE.SUB.

Digital Research provided a number of customising programs but Locomotive Software created several more to enable the 6128 machines to be configured for a range of users. These are not standard CP/M commands; they were added by Locomotive Software and will only work on Amstrad machines. As far as I am aware, these Locomotive Software programs are not PD, unlike the programs created by Digital Research.

If you want to use dates under CP/M Plus, the first command you would normally put in PROFILE.SUB is Digital Research's `DATE SET` (or John Elliott's `XDATE SET`) which will run `DATE.COM` (or `XDATE.COM`) and prompt you to enter the date. Then, if you want to use menus, you need to add

```
SETDEF [order=(SUB,COM)]
```

and you may also want to use

```
SETDEF b:,*
```

to ensure that CP/M searches both drives starting with drive `b:` for any files. Then, if you have an unconventional setup — a serial printer, for example — you can use `DEVICE.COM` to set it up.

Next you would normally put in Locomotive Software's offerings.

PALETTE.COM allows you to choose a background and a foreground colour within the range 0–63 with 0 being black and 63 being bright white; since the CPC cannot reproduce 63 colours, you will find repetitions in the sequence. `PALETTE 16, 51` produces a bluish white on a green background. `PALETTE 63, 0` produces a black on white screen on Plus machines and colour monitors and a black on green screen on the older mono monitors.

LANGUAGE.COM allows you to choose which of the first 8 ISO languages will appear on screen — ISO 3 is English with pound signs; so `LANGUAGE 3` makes pound signs appear.

SETSIO.COM takes a series of parameters to set up the serial interface; however, unless you are trying to communicate with an unusual terminal, `SETSIO 4800` or `SETSIO 2400` (to set the baud rate to 4800 or 2400) is usually enough as the default settings for all the other parameters are fairly standard. Whenever you run PROFILE.SUB with `SETSIO 4800` or `SETSIO 2400` in it, `SETSIO.COM` will also list all the other parameters which it has set automatically.

SETKEYS.COM and SETLST.COM set up the keyboard and printer respectively. They both need a separate file containing the keyboard or printer definitions. KEYS.WP supplied with 6128s contains all the definitions needed to run TASWORD and some other word processors. KEYS.CCP contains the standard CP/M definitions but you can ignore these unless you are into hacking CP/M programs. My more ambitious key definition file for use with Supercalc 2 was published in WACCI 79. Printer definition files are constructed in a similar fashion.

Putting all these regularly used commands in PROFILE.SUB means I can switch on and be into my word-processor in less than 90 seconds — which is about the same time as it takes Windows 95 to sort itself out and let you get into your word-processor. Funny how a computer running at 50 times the speed of a CPC still takes about 90 seconds to let you start working!

Changing profile

You can change the keyboard settings under CP/M 2.2 by using several discs each set up differently using `SETUP.COM`, one for your wordprocessor, one for your database and one for your spreadsheet. This works well enough since you always have to have a system disc in drive a: when using CP/M 2.2. However, once you have loaded CP/M Plus and configured your machine using the commands in `PROFILE.SUB`, you can put away your system disc and work entirely from data discs.

So I keep shorter .SUB files for my word-processor, database and spreadsheet because usually the only things I need to change when I swap from one to the other are the keyboard settings. So VDE.SUB consists of

```
SETKEYS KEYS.VDE
VDE266
```

while SC2.SUB contains

```
SC2DATE
SETKEYS KEYS.SC2
SC2.COM JRHUDSON
```

Entering VDE or SC2 (or even VDE.SUB or SC2.SUB if you have not used `SETDEF [order=(SUB,COM)]`) is a lot quicker than typing everything in every time and having to remember the '266' in VDE266! `SC2.COM` has to be written out in full in the .SUB file if you have used `SETDEF` so that CP/M doesn't call `SC2.SUB` again and interpret `JRHUDSON` as a parameter of the .SUB file. If you haven't used `SETDEF`, you can leave '.COM' out of the last line.

Supercalc scripts

Supercalc2 has its own scripting language for which you have to store scripts in .XQT files. So I have a series of Supercalc commands in JRHUDSON.XQT which load a particular spreadsheet and carry out various operations automatically. By including `JRHUDSON` after `SC2.COM` I can move smoothly from CP/M's scripting language into Supercalc's. But Supercalc is fussy about quitting to CP/M and will only quit properly to a file it can find on disc. If you add another CP/M command after a line like `SC2.COM JRHUDSON` and quit normally from SC2, CP/M will take over straight away. If a line like `SC2.COM JRHUDSON` is the last one in a CP/M script, you need to use 'Quit To' in SC2 and give the name of a CP/M script or program on the same drive as SC2 for SC2 to quit without a fuss and for automatic progress to continue.

Boring jobs and parameters

CP/M SCRIPT really comes into its own with repeated commands such as those used in jobs like debugging or backing up. If you are developing a program in assembler or C, you normally enter the source code, compile the program and run `HEXCOM` before testing the program. So creating a program in assembler might involve:

```
WP NEWPROG ; to write the source code
MAC NEWPROG ; assemble the source code
HEXCOM NEWPROG ; convert from Hex to Binary
NEWPROG ; run the program
```

or, in CP/M 2.2

```
WP NEWPROG ; to write the source code
ASM NEWPROG ; assemble the source code
LOAD NEWPROG ; convert from Hex to Binary
NEWPROG ; run the program
```

It would be rather tedious writing a new script with the name of the program in every line; so CP/M allows you to add parameters. You can create a file perhaps called CREATE.SUB:

```
WP $1
ASM $1
LOAD $1
$1
```

and, whenever you need it, you simply add the name of the file you are creating, for example, `CREATE NEWPROG` (or `SUBMIT CREATE NEWPROG` if you have not used `SETDEF`). CP/M will then replace every `$1` with `NEWPROG`. You have to say `$1` because you can add several parameters. For example, if you find `PIP` confusing, you can create a script called COPY.SUB with the line

```
PIP $2=$1[v]
```

Then, if you have used `SETDEF` as described earlier in this series, when you enter

```
COPY A:MYFILE B:MYFILE
```

CP/M will copy A:MYFILE to B:MYFILE

If you are using CP/M 2.2 or you haven't used `SETDEF`, you have to say `SUBMIT COPY` . . . or `COPY.SUB` . . .

Backing up

If you have used `PIP` in multiple command mode, you will know that it will put up an asterisk and allow you to enter a sequence of commands. If you want to use `PIP` in a .SUB file, you can put the same commands in prefaced by '`<`'.

So, for example, I happen to have some databases which are split between Europe, America and the rest of the world. The file AMERICA.SUB contains the lines

```
PIP
<a:=b:america.dbf[ov]
<a:=b:america.ndx[ov]
<
```

The last '`<`' is needed because PIP exits multiple command mode with a simple `<RETURN>`. So, all I have to do to back up my American database is to put a disc in A: and enter AMERICA.SUB from B:

From time to time I make changes to the programs which I have written to use these databases and I back these up on a separate disc using a short script called BAKDBASE.SUB:

```
PIP
<a:=b:*.cmd[av]
<a:=b:*.fmt[av]
<a:=b:*.frm[av]
<
```

In this case, I am only interested in the files which I have changed; so I add the `[a]` or 'archive' option.

The '`<`' works for both CP/M Plus and CP/M 2.2 users, the difference being that you have to leave out the `[a]` or 'archive' option, add the line `XSUB` at the start of the .SUB file in CP/M 2.2 and have `XSUB.COM` on the disc along with `SUMBIT.COM` and `PIP.COM`. Also, without the 'archive' option, you have to copy all the files; whether that would take longer than entering each file name separately would depend on how many program files you had written or changed.

Total control

This ability to accept commands for execution within a program can be extended to any standard set of instructions. To take another example, if you want to use dates or passwords with CP/M Plus, you have to use `INITDIR` after `DISCKIT3` in order to prepare the discs for dates and passwords. Digital Research made this a separate process so that people could carry on using discs without dates and passwords with CP/M 2.2.

But `INITDIR.COM` always asks you whether you want to proceed and does not offer you the normal response at the end of a formating program '`Format another (Y/N)?`' As I usually format discs in batches of ten, I wrote a short .SUB file to simplify the process:

```
INITDIR.COM B:
<Y
SET B:[ACCESS=ON,UPDATE=ON
; Put another disc in drive b: and
: press a key to continue or
; press CTL-C to end
PAUSE
;
;
;
INITDIR
```

As before, if you are using `SETDEF [order=(SUB,COM)]`, you have to add ".COM" to `INITDIR` in the first line; otherwise you end up in a loop. If you are not, you can leave ".COM" out but you need to change the last line to INITDIR.SUB.

When `INITDIR` asks you whether you want to proceed, CP/M sends '`Y`' to the program as if it had been keyboard input.

`SET` initialises the two forms of the date I want to use and `PAUSE.COM` gives me time to put another disc in before the whole sequence starts all over again.

Pausing

I put a few comment lines in after `PAUSE` because `CTL-C` does not operate instantly; `PAUSE.COM` intercepts the first `CTL-C`; so you have to hold it down until `PAUSE.COM` has returned control to the special temporary file which then halts in response to `CTL-C`.

If you don't have `PAUSE.COM` or `WAIT.COM` or the GRADUATE CP/M ROMs, you can create a pause with `ERA B:*.*` because this will always put up the prompt '`ERASE B:*.* (Y/N)?`' which will not cause any problems since you are dealing with blank discs. You can then change the disc and answer '`N`'. Alternatively, you can get hold of the public domain `IF` commands which include options to pause and quit.

Instead of using '`<`' in CP/M Plus, you can use `GET.COM` and put `GET MYFILE` before `INITDIR` where MYFILE contains a list of instructions you want the program to act on. In this case, there isn't any point as there is only one instruction but, if you have a long series of instructions, it may be better to put them in a separate file and edit that when you want to change them rather than editing the .SUB file every time.

Recap and more

To round off this chapter, I will recap the main features of CP/M SCRIPT and look at `IF.COM` and a point about user areas.

You can automate a range of repetitive or complicated tasks in CP/M using features which were built into CP/M from Version 2. A few programs make direct use of these features but

Digital Research also provided a program called `SUBMIT.COM` which enables you to convert a sequence of instructions into a temporary file which CP/M will execute.

To do this, you put your instructions in a file, perhaps called COPY.SUB, and enter `SUBMIT COPY` or `COPY.SUB` or `COPY` if you have used `SETDEF` in CP/M Plus.

If you want to use the same instructions with different data or program files, you can use `$1`, `$2`, etc. to stand for each parameter in the sequence of instructions in `COPY.SUB` and give their names each time you run the program, for example, `SUBMIT COPY OLDFILE NEWFILE`.

If you want CP/M to pass instructions to a program, you preface these with '`<`' (and run `XSUB.COM` in CP/M 2.2). Alternatively, in CP/M Plus, you can use `GET.COM` to pass a series of instructions to a program.

CP/M Plus comes with an additional feature; when it is loaded, it always tries to find and run PROFILE.SUB; you can use this feature to fully automate the way CP/M Plus runs. Locomotive Software added a range of Amstrad specific programs to make this easier for 6128 users.

In addition, there are a number of PD programs which extend CP/M SCRIPT, such as `PAUSE.COM`, `CLS.COM`, `SKIPIF.COM`, `QUITIF.COM` and `IF.COM` of which the last three deserve special mention.

IF.COM

`IF.COM` takes advantage of another feature of CP/M which few people have exploited — error codes. If you put `:` at the start of a line in a CP/M script it will only execute it if there is no error code. So you could have a script called LETTER.SUB:

```
IF / $1                 ;if $1 does not exist
:PIP $1=TEMPLATE.LET[v] ;create $1
WP $1                   ;load word processor and edit $1
```

When you add the name of a file after LETTER.SUB, CP/M Plus will look for the file and, if it does not find it, create it from your letter template before loading your wordprocessor. Otherwise it will load the wordprocessor immediately.

The same script would work slightly differently in CP/M 2.2 where the PD programs `SKIPIF` and `QUITIF` do not rely on error codes but simply skip a line if something is true.

```
SKIPIF E $1             ;skip the next line if $1 exists
PIP $1=TEMPLATE.LET[v]  ;create $1
WP $1                   ;load wordprocessor and edit $1
```

Both the enhancements of `SUBMIT.COM` for CP/M 2.2 and `IF.COM` for CP/M Plus extend the capabilities to CP/M SCRIPT significantly. Version 3.4 of `IF.COM` allows a wide range of conditionals including a simple pause condition (so you can dispense with `PAUSE.COM`) and can be patched onto `SUBMIT.COM` so that it is there every time you use `SUBMIT.COM`. In this case, you begin the relevant lines in your .SUB file with '`?`' rather than '`IF`'. So you have to decide whether you are going to patch `SUBMIT.COM` or not and write all your .SUB files to suit whether you are using `IF.COM` as a standalone program or through a patched version of `SUBMIT.COM`.

User areas

A CP/M script, like a program, can use one user area and the system area. But there is one quirk in the way CP/M handles files in the system area which can create problems with free-standing CP/M spellcheckers. When using these, you usually need to exit the wordprocessor, use the spellchecker and then return to the wordprocessor to reformat the text.

If you make all the spellchecker's files, including the custom dictionary, [sys] files in area 0, you can run the program from any user area but, when it has finished, the spellchecker will write the new custom dictionary to the current user area and not to user 0 and it will lose its [sys] attribute. So you can end up with a different version of the custom dictionary in every user area (and the disc rapidly fills up!).

To get round this problem, I wrote the following script:

```
SPELL $1
SET CUSTOM.DCT[sys]
PIP CUSTOM.DCT[g0]=CUSTOM.DCT[rv]
ERA CUSTOM.DCT
```

which I only use outside user 0. This spellchecks the document, makes the new custom dictionary a [sys] file in the current user area and then copies it back to user 0 over the original. The 'r' in `[rv]` is essential as `PIP` will only copy [sys] files if you add the 'r'. Finally, it erases the custom dictionary from the current user area.

Obviously, this script must not be used in user 0 as you will get an error message in line 3 and a disaster in line 4! So I keep a separate script for spellchecking documents in user 0.

4 The wonders of PIP[1]

The tremendous popularity of NSWP shows how difficult it is to cast off a reputation. PIP Version 3 has virtually everything that NSWP has and more; but NSWP was developed from SWEEP which attempted to overcome the limitations of PIP Version 2.

So often people write into magazines with queries which `PIP`, or other CP/M Plus utilities, could solve for them, I thought it might be useful to provide a run down of the many features of `PIP` which I use and which are there for free to 6128 owners. I'll also cover the differences with `PIP` version 2.

`PIP` is often criticised as a copying program but that was only incidental to its original purpose as the Peripheral Interchange Program. It provided an easy way of linking the different peripherals which made up the early micro-computers. It drew on the syntax of the day in that assignments in BASIC were made with `LET x=y`; so assignments in `PIP` follow the same format. `PIP a:=b:*.*` means 'LET drive A have all the files on drive B'.

I say 'drive' rather than 'disc' above because PIP follows the convention that where you get something from or send something to ends in a colon. So the monitor is `CONOUT:`, the keyboard is `CONIN:` — usually combined as `CON:`, the LiSTing device (printer to you young 'uns) is `LST:` and so on. So '`A:`' and '`B:`' mean drive A and drive B to `PIP` — it doesn't bother which disc is in the drive unless CP/M reports a problem.

`PIP` Version 2 has four assignments other than drives: `CON:`, `LST:`, `PUN:` and `RDR:`. `PUN:` is for a serial output device, or PUNch; `RDR:` is for a serial input device or ReaDeR. So 464/664 users can use a printer and a serial port with CP/M.

By the time Version 3 came out this was too limiting and Version 3 can have as many separate devices as you want — the only limitation being that each must have a unique name. On the 6128, they are `CON:`, `LST:` and `AUX:`, which is the serial port, and `CON:` and `AUX:` can be split into `CONIN:`, `CONOUT:`, `AUXIN:` and `AUXOUT:` if required — the latter is essential for 1200/75 BAUD transmissions. PCW users will know that `CEN:` is the add-on external printer port and `PAR:` the built-in external printer port.

If I want to check what is in a text file `PIP CON:=D:filename.ext[z]` will send the file to the screen — see `[z]` on page 23 for an explanation — though this is not recommended for a long file as you cannot stop `PIP` once it has started; I usually know what is in long files — it's the short files I forget.

If I want to write a short text file such as a .SUB file, `PIP D:filename.ext=CON:` puts the cursor at the left margin of the screen. I then enter the relevant text. You cannot use `<RETURN>` in text files created with PIP as `<RETURN>` replaces two characters in most files; you have to use `CTL-M` followed immediately by `CTL-J` to create the two characters for PIP. Similarly, you cannot use `DEL` or CLR. So you have to avoid making typing errors — but I usually manage to do that for the short files for which I use `PIP`. At the end, you have to add the CP/M end-of-file marker `CTL-Z` at which PIP will whir into action and copy the text to disc.

[1]Originally submitted to WACCI on 12 February 1994

Transferring files

I use `PIP D:filename.ext=AUX:` to receive files from the Notepad. Unfortunately, the Notepad was designed with PCs in mind and does not send `CTL-Z`. However, if you pull down the special characters with Symbol-Menu and enter the right arrow symbol two lines above Z in the display, this is in fact `CTL-Z` and `PIP` will whir into action as soon as it receives it. I don't put `CTL-Z` into each file; I have a separate file called `CTL-Z` on the Notepad and send that after each file I transmit. But whichever way you do it, you don't need any fancy software to download files from the Notepad, just a serial interface and a cable. After experimentation, I find 4800 BAUD is quite safe with the Amstrad/PACE serial interface — 9600 is too fast.

Incidentally, my .SUB file only contains the lines:

```
SETSIO 4800
PIP
```

as everything else is already correctly set up on the 6128s.

With the 464/664 you need to set up the serial interface differently and use `PIP D:filename.ext=RDR:` but otherwise `PIP` works in exactly the same way.

As `PIP` holds the file in memory before passing it on, there is a limit on the effective file size when receiving material using the serial interface — though I have transferred files of over 30K using version 3 — since there comes a point when `PIP` has to write some of the contents of memory to disc during which incoming characters may be missed. However, as long as the receiving device can handle it, there is no limit on the length of files `PIP` can transmit.

Printing files

You can use `PIP LST:=D:filename.ext` to print any text file; if it is not in ASCII format add `[z]` to the end of the command and if you use a sheet feeder rather than continuous paper add `[p]` to ensure `PIP` makes a page break.

Finally in this part, you can use `PIP LST:=AUX:` with the Notepad to print short files directly via the serial port rather than unplugging your printer from the computer. You need to have everything ready with the print head at the top of form or you will lose some of the early characters and you cannot exceed the printer's buffer because `PIP` is a one way operation — it cannot return any commands from the printer to the Notepad to stop sending characters. So a slow BAUD rate is recommended for the serial link. To unhook the connection, just send a `CTL-Z` from the Notepad — it doesn't affect the printing but it lets `PIP` know you have finished.

Multiple transfer

Though all the examples I have given show the command tail following `PIP`, you can also use `PIP` in multiple transfer mode by entering `PIP<RETURN>`. An asterisk appears on the left hand side of the screen. You can then remove the disc with `PIP` on, leaving that drive free for copying between any other discs and you can enter as many command tails as you wish until you have finished with PIP. You then press `<RETURN>` and `PIP` exits to the CP/M prompt.

If you are repeating a command such as `a:=b:*.*[aov]` when you are backing up a load of discs, you can use the CP/M command `CTL-W` to recall the last command.

You may also find it useful in a long session to close the read only hole on the discs you are copying from because `PIP` cannot warn you if you mix them up. However, it will tell you if you put a read only disc in the destination drive, allowing you to swap discs and use `CTL-W` to repeat the instruction.

As will be clear already PIP can take a whole range of additional commands which affect the material being transferred in one way or another. For a full list, consult one of the CP/M text books; the following are the ones I use from time to time.

[a] is only available in Version 3 and sets the archive bit on the file. Whenever you use `[a]`, `PIP` ignores files with the archive bit set and sets it on every file it copies successfully. When programs next use the file, they write the file name to disc without the archive bit set. So `PIP` always knows which files are new versions. This works with CP/M 2.2 and AMSDOS files so you can back-up Mini-Office files, for example, on 6128s using `[a]`. The archive, read only and system bits are the eighth bits on the file extension. CP/M only shows seven bit characters on screen but Supercalc2, and some other CP/M programs, show all eight bits which explains why a disc directory in Supercalc may look a little odd. The archive bit is also useful if you find corruption on a disc. `PIP` will stop copying if it finds corruption and it will also tell you the name of the file in which it occurs. You can then use `SET D:filename.ext[archive=on]` to mark the corrupt file and `PIP` will skip this file and copy all the sound ones to a fresh disc if you use `[a]`.

[c] another Version 3 facility — lets you confirm each copy. Rather than entering individual file names, if you use `PIP a:=b:*.*[c]`, `PIP` will list all the files on the drive and ask you to say which ones you want transferring. You can restrict the ones that are listed and then transferred with *.BAS or *.DOC or prog*.* if you want. This is DR's answer to tagging in `NSWP` and can be quicker in some circumstances and slower in others.

[g] followed by a number instructs `PIP` to look for or add the relevant user numbers. `[g]` stands for GET because USERs were originally an MP/M facility which CP/M 2.2 cannot use. So, for compatibility, DR needed a way to get files from an MP/M disc into use under CP/M 2.2. `PIP b:=b:filename.ext[g5]` gets the file in USER 5's area and copies it into user area 0 where CP/M 2.2 can use it. There is no 'move' command in PIP because 'moving' the file would mean USER 5 would not be able to use it under MP/M. You cannot copy the file back under Version 2 but in Version 3 you have direct access to all user areas under CP/M 3.1. However, we are stuck with `[g]` for compatibility.

[l] and [u] I only use in one context — transferring sound commands between BASIC and LOGO; on the Amstrad these commands are identical except that they are in upper case in BASIC and lower case in LOGO. `[l]` not unsurprisingly turns all upper case characters to lower case and `[u]` does the opposite. In practice, you need to make an ASCII save in BASIC and then a find and exchange with a word processor to get the BASIC commands into LOGO format and you could use the same find and exchange within a word processor to convert the case but PIP is much quicker with a long sequence of commands.

[o] stands for object file. As I mentioned above, `PIP` looks for `CTL-Z` as an end-of-file marker but this could be a legitimate character in a binary file. `[o]` tells PIP to ignore `CTL-Z` and copy to the end of the last sector mentioned in the disc directory. It should be used with .COM, .BIN, .CAL and .DBF files.

[p] is mainly useful for setting the page length when using single sheet feeders with `LST:`; the default is 60 lines but you can change this by adding a different figure after the `p`.

[r] stands for read system files. `PIP` ignores system files as it assumes you will normally only be transferring data files. So you need to add `[r]` to tell it you really do want to transfer a system file.

[v] stands for verify and PIP will check what it has written to disc against the copy it has in memory (not against the original). Though this may seem to be a problem, I have found this sufficient when copying between a silicon disc and a floppy disc where the differing speeds of the two media has caused problems for other file copying programs, particularly on long files.

[w] stands for write over a read only file. `PIP` assumes that you do not want to over-write a read only file and gives you the option of stopping the transfer every time it encounters a read only file with the same name; you can stop it throwing up the prompt every time it encounters a read only file with `[w]`.

[z] zeros the eighth bit; in other words, it creates an ASCII compatible file. As some word processors use the ASCII characters below 32, it does not strip out these control codes but it allows you to display text files on screen without any confusing characters. I have also used it to create ASCII files for transmission to others. But in this case, it is important to remove all control codes from the original document first and to save it in ragged right format. Otherwise PIP will create a right-justified file which is a pain to reformat on any other word processor!

Finally `PIP` Version 3 recognises passwords created under CP/M 3.1 and will not copy a protected file without the password. You add the password with ;password after the file name before any additional commands and `PIP` copies the password and sets the protection level to READ on the copy PROVIDED the destination disc is already set up for passwords. `PIP` does not warn you if you are copying a protected file to an unprotected disc. I cannot think of any way of preventing potential confusion between source and destination discs but I think the programmers could have put in a warning similar to the one with read only files to warn people if a protected file would not be protected following transfer.

Otherwise, as you will see, I find `PIP` Version 3 a satisfactory and remarkably versatile program for daily use and even Version 2 has a collection of facilities not elsewhere available in a single program.