

**WAGEI**

138

Contents...

**01 - Thanx & Stuff**      **02 - Fair Comment**      **03 - A to Z of the CPC**      **04 - From the Archives**  
**05 - Twenty Questions**      **06 - Keyboard Scanning**      **07 - Meet the Relatives**      **08 - Programmers' Patch**  
**09 - Genesis**      **10 - Famous Last Words**



## Thanx & Stuff

### The Late Western Express

Sorry, sorry, I've missed the two-month deadline again. The last contributions have only just trickled in, but it's arguably my fault, I s'pose, for not having the courage to put out an eight-page WACCI when that was all I had after two months. Never mind. Hope you like it anyway.

### I'm on the train

I've just bought a new CPC - sort of. It's a rather swish Apple iBook, a lightweight portable Macintosh with DVD player and all that sort of thing. It runs Quark XPress, so I can edit WACCI on the move, and the CPC emulator Arnold - so I can program on the move.

Matthew's Programmer's Patch article last issue was especially timely. I read it, thought "ooh, I like the sound of that Zmac", and got hold of a copy. So now I have a portable CPC development system, and two hours' commuting time each day in which to use it. You never know, I might even finish ChaRleyTraCker/Palatine/BTL 5/Fidelity (delete as applicable).

I've also replaced the monitor on my desktop Mac (see last issue's Thanx & Stuff), which makes laying WACCI out a bit easier. Sorry. I'm sure you don't really want to know all this.

### Here is a news

A real news, as well. Amstrad has just announced they're restarting production of one of their 8-bit computers. Not the CPC, sadly, but the Spectrum. No, I'm not making this up.

Many of you will be familiar with the Amstrad E-m@iler - a glorified telephone with a QWERTY keyboard and Internet capabilities. Aimed at people who'd like to contact friends electronically without the hassle of a PC, it enables you to write and receive e-mails for an outlay of about £100.

The first version, I thought, was a bit simple - it could have included a web browser, for example. But Amstrad have just relaunched it, and the new E-m@iler is... interesting.

Not only does it include a very simple browser (Microsoft Mobile Explorer), there's a built-in Spectrum emulator. Should you feel like a quick round of Chuckie Egg, you just connect to an Amstrad website via your E-m@iler, download the game, and start playing.

Quite astonishing, though not completely unprecedented - I think there was a digital TV set-top box recently that could emulate the Commodore 64, and the C64's fabled SID sound chip has even been repackaged as a MIDI synthesiser for professional electronic musicians.

And I'm not 100% convinced that the new E-m@iler is very well-thought out. The E-m@iler's target market is largely technology-hostile, and unlikely to be swayed by the promise of Spectrum games. Retro computing enthusiasts, meanwhile, are probably already using a Spectrum emulator

on a PC. But time will tell.

Incidentally, why not the CPC? There's one good reason - the kitsch factor. The Sinclair ZX Spectrum is a byword for all things '80s and retro in the way that the altogether more competent CPC will never be. By way of illustration, I'm happily writing here about Amstrad reviving the Spectrum. I doubt that a Spectrum fanzine would pass comment on Amstrad reviving the CPC.

Should any of you buy the new E-m@iler, though, do write in and tell us all what it's like.

### **You're all very clever**

Or at least, that seems to be what we're unintentionally assuming at the moment. Most of the past couple of WACCIs have been taken up with techy articles. Which I'm not, personally, complaining about - I love reading that sort of stuff.

But I'm quite conscious that lots of you do other things with your CPC, and WACCI should appeal to you, too. So a renewed plea. If you'd like more articles on Protext, games, CP/M, BASIC, or whatever, tell us. Better still, suggest what you'd like to read about; best of all, write something yourself. Thank you.

All of which leads to the big question: the CPC - retro or current? Do we write about the CPC as a fun toy to play around with, an enjoyable relic from history - or as a useful little electronic box that can still fulfil modest computing needs? I don't think the two are necessarily incompatible. But this, more than anything else, will shape future WACCIs.

### **PdR - International Man of Mystery**

It would usually be time to hand you over to your new old Fair Comment editor, Mr Philip DiRichleau, but it's Brian Watson this month. Philip has recently got married and had another kid, meaning that there are now five people in Lincoln with an unspellable surname. He's also been ill.

That notwithstanding, he would love to receive your letters. Peter Rogerson, where are you?

*Richard 'CRTC' Fairhurst*

Editorial address: 5 Nine Acres Close, Charlbury, Oxfordshire OX7 3RD. [richard@systemeD.net](mailto:richard@systemeD.net)

Admin enquiries: 'Number Six', Windmill Walk, Sutton, Ely, Cambs CB6 2HN. 01353 777006



# Fair Comment

*with Brian 'Loonybins' Watson*

## Oh no, not him again...

*Why is that Brian Watson bloke editing Fair Comment?*

*Mr A Reader  
Birkenhore*

Because no-one else is. Philip DiRichleau is the Fair Comment Editor, but he's either otherwise occupied or the Great WACCI Readership has not sent him anything to Edit. I believe a combination of the two is the most likely reason. [He's been ill - Richard]

## A matter of convention

*What are the plans for a WACCI convention in 2002?*

*Mr E Party-Animal  
Welwyn, Herts (where? - Richard)*

As yet, none, but it's a perfect topic on which to write to WACCI to express your opinion. Letters received now, before decisions are taken, will carry more weight than the usual round of "I can't afford ten quid for a WACCI get-together as I've I've just bought a thousand quid's worth of PC equipment", "why not hold it five minutes from me here in Orkney?", "have you any idea what rail travel costs these days?" letters that generally arrive once a venue has been announced.

What is happening is a [comp.sys.amstrad.8bit](http://comp.sys.amstrad.8bit) newsgroup get-together in a pub in London (at the latest estimate) quite soon. If you fancy the idea of something less highly-organised than the grand conventions that Angie has arranged in past years (either instead of, or as well as, something along the lines of previous years) feel free to express your preferences in the afore-mentioned letters. Philip would love to hear from you.

## ...and again, and again

*Why does Brian Watson seem to be the only one doing anything towards running WACCI? Has he bought a controlling interest in WACCI Holdings shares? He is the Treasurer, Publicity Officer, he writes a lot of the articles (eventually - Richard) and now he's got his mitts on Fair Comment? Is their no end to this megalomaniac's hold on the CPC community, and WACCI in particular?*

*Mr Angry  
Barrettholme, W Yorkshire*

I'm not the only one doing anything. Richard's editing the magazine (and he's the best Editor WACCI's ever had, in my opinion). Angie Hardwick keeps the subscription lists up to date (a fiddly job that I'm glad I don't do) and is a "first-line receiver" - like our PD librarian Ray Powell, to whom I urgently owe a letter and some blank discs - of WACCI monies from the membership and they act as a check that I (as the WACCI Treasurer) am not embezzling it.

As a member of the Committee, Angie is sent periodic statements of the WACCI cashflow with photocopies of our bank statements to confirm the figures I record in our accounts books. Frank Neatherway is our Chairman, through which all organisational suggestions are implemented and

he's no pushover, I can assure you! He has a casting vote in the event of a split vote between the other Committee members on any matters of WACCI business.

As far as the Publicity hat is concerned, I do it because I'm good at it, I'm passionate about WACCI and the CPC, and I am well-connected, in both a CPC and an electronic sense. Publicity is not really a one-person job though, and anyone who would like to share the load is welcome to volunteer.

As to writing a lot of the articles in WACCI, I think I'm good at that too. If that sounds like arrogance I suggest, Mr Angry, that you set fingers to keyboard too, because in every issue there are acres of white space waiting to be filled, just like this one, and every issue includes an appeal for written contributions that is largely ignored.

That's no big deal as far as I'm concerned because I understand that in any club without paid officers, 2% of the membership always do 98% of the work, but I'm not then going to apologise for getting stuck in when other people feel that they cannot.

As to what I write, I have to write from the ordinary CPC user's point of view because I have no idea what happens "under the bonnet" of my computer. If that is your point of view as well, it is as valid here as the more technical articles that are received from those able to write them.

## **An accountant writes**

*How much has WACCI got in the bank?*

*Little Miss Nosey*

*Sennor*

As of 31st December 2001, £1325.57. What that is in Euros, I have no idea.

What I do know is that it is enough money to fully repay any/all WACCI subscribers - plus a small surplus - should they all demand a refund on the same day.

## **Return of the slo-bot**

*I have just heard that geezer Brian Watson will be Editing Fair Comment in this issue of WACCI. It gives me possibly my only chance to ask him in print and in public this question, "Why do you take so long to answer letters, copy discs, handle phone enquiries, etc, Mr I-do-everything Watson?"*

*Mr JRR Tolkien (No, not that one)*

*Hobbiton-Under-Hill, The Shire*

*(Damn! What a giveaway!)*

In short, because I have a life.

In long (if there is such an expression), like all the other people who do their bit to ensure you get this magazine hitting your mat from time to time, I do my best for WACCI and the wider 8-bit community within the time that is available to me. It has to compete with time for work, for family life (such as that is) and whatever other mad passion I may be following at the moment. Sorry, but that's the way it is.

[And now for a real letter - Richard]

## **Goodbyeeee...**

*Dear All and Members of WACCI,*

*I am at a turning point right now. I feel a waste to WACCI, as there is nothing doing or a foot. I wish to resign from the committee and leave this worthless post to someone else.*

*I for one have found nothing new in the magazine. I have said for a long while now The Club should be closed down.*

*Angie our original idea was to rescue the Club and make it solvent so all could be paid back.*

*I feel that has been achieved, therefore I see no Purpose in delaying the inevitable.*

*I wish to each and everyone of you, all the best and thanks for all the help and companionship*

*through all the years I've been a member. The Club has given me some great friendships over the years and they will remain with me always.*

*Please Print this in the Magazine as written.*

*Christine Raisin*

*Nottingham*

First of all, Christine, an apology if this isn't 'as written'. Since you didn't send it to Philip (Fair Comment ed) or myself (Richard), we have no way of telling - we only received it as a second-hand e-mail with someone else's comments interpolated, and potentially some of your bits snipped out.

Secondly, a genuine thank you for all your work for WACCI over the years. (I remember the first time I came into contact with you, when you asked me to debug a Mastertronic game that had never quite worked - Mindtrap, I think? - and subsequently meeting you at one of those wonderful Bescot conventions. Debugging the game was good fun, too.)

Thirdly, I am sorry to see the tiresome question of dissolving WACCI come up again. In my experience, it's only ever committee members who have moved on from the CPC who suggest this.

WACCI is, and always has been, a CPC magazine. Its existence as a 'club' derives solely from the community of CPC users who know each other through the magazine. Therefore as long as there's someone to edit the magazine, WACCI will continue.

You can't just send 150 people's subscriptions back, Christine, simply because you're no longer interested in CPCs. 150 people are interested enough to pay their subscriptions. As Brian has pointed out, if they're not, all they have to do is write to him and ask for their money back. - Richard

### **...and thank you**

*Someone has kindly sent me a sample copy of WACCI issue 137 and I must congratulate everybody involved with this new-look CPC magazine. Up until now, PCW Today has taken the plaudits for the best ever non news-stand 8-bit publication but I would say it now has a serious rival in terms of quality (and content) although of course it caters for a different market. Additionally, WACCI is published bimonthly which is a schedule PCW Today readers can only dream about. [...]*

*Steve Denson*

*SD Microsystems/LoCoScript Software*

This is an excerpt from a posting to the [comp.sys.amstrad.8bit](http://comp.sys.amstrad.8bit) Internet newsgroup rather than a real letter - hope you don't mind it being reproduced, Steve - and perhaps it balances the last letter out a bit. It might even embarrass me into getting WACCI 139 out on time. :-), as they say on the Internet. - Richard

Send your letters to Philip diRichleau, 32 Arboretum Avenue, Lincoln LN2 5JE, or e-mail [philip.dirichleau@ntlworld.com](mailto:philip.dirichleau@ntlworld.com).



# The A-Z of the CPC: CP/M

*Your CPC's alternative operating system is the gateway to a whole world of free software, as David Cantrell explains*

## Don't get a complex

Your CPC is a sufficiently complex computer that it needs an operating system. This is a program which runs when the machine starts up which provides basic functionality like making it easy to use the hardware.

For example, it provides 'firmware' functions which let you put text on the screen. Without those, you would have to access the screen memory directly, and that's a bitch of a job as any demo coder will tell you. Normally, the CPC runs a simple ROM-based operating system which includes the Locomotive BASIC interpreter.

If you have a disk drive, then one of the things the OS does when starting up is to run a small program in the disk ROM. This program 'patches' the OS to add support for the disk. It adds a few RSXes and changes the code in memory to replace the OSes cassette support with disk support. This patched OS is known as AMSDOS.

## The old world

But if you have a disk drive, then you will also have another OS available to you - CP/M. This OS was very popular indeed in the late '70s and early '80s for microcomputers. Its popularity came from the way it hid the details of the computer's design from the programmer, and that it did this job on very many different computers. Back then, there was no standard hardware design like we have today with PCs.

Instead, all micros were radically different. They all accessed their disks differently, wrote to the screen differently, read data from the keyboard differently. CP/M hid all that.

Any programmer working in CP/M knew that his program would work on any other CP/M computer, regardless of whether it was made by Northstar, Kaypro, Amstrad, or any of a hundred other now-defunct companies. Of course, for all this magic to work, each computer manufacturer had to write their own special version of CP/M to handle their unique hardware. But as far as the users and programmers were concerned, there were no differences - because CP/M made things Just Work.

## Boot camp

Unlike AMSDOS, where the OS is stored in ROM, CP/M is stored on disk. On all computers, including the CPC, the very first thing the computer does is run a program stored in ROM. On the CPC, this program is the built-in OS.

However, on most other machines, including the Amstrad PCW and the IBM PC, that program in ROM is much simpler. It knows just enough to read the first sector of a disk and then execute whatever it finds there. On the CPC, the equivalent of this titchy ROM program is the |CPM command, which is one of the RSXes set up by AMSDOS.

This tiny program, usually just 512 bytes long, is known as a 'boot loader'. It is a little more clever than the program in ROM; it knows enough to read another program from the disk and run it. In the case of a CP/M machine, that program is CP/M itself. (Things are a bit more complicated with the later versions of CP/M, but let's ignore that.)

## **Circles within circles**

CP/M is just an operating system, nothing more, nothing less. On its own, it is useless. It would be like the Mac OS with the menus, icons and pointer removed; or Linux without the ability to type in a command; or, come to think of it, AMSDOS without Locomotive BASIC.

So the very first thing CP/M does is to load a program called the Command Control Processor, or CCP. This is a special program, and is treated somewhat differently to all other programs which are stored as files on the disk. The CCP, on the other hand, is so intimately tied to CP/M itself that it doesn't appear as a file on disk. Instead, it lives in the 'reserved' area of disk along with the OS. And what it does is present you with the familiar A-prompt.

## **What it feels like for a CPC**

For programmers, CP/M provides a set of calls which do pretty much the same job as some of the CPC firmware calls do - print text, load data from disc, that sort of thing. Only a limited range of features are available, however. Since CP/M programs have to work on all CP/M machines, it only provides the functionality available on every such machine.

That means no graphics, for instance, and no sound beyond a simple beep. It is technically possible to get at the CPC's other functionality, but this is frowned upon, as it will make your code incompatible with every other CP/M machine in the entire universe - rather defeating the point of writing a CP/M. program.

Whilst the CP/M calls provide much the same functionality as the firmware, the calls themselves are rather different. They are at different memory locations, the input and output registers differ - the programming interface really is very different. But the concept is the same.

Much CP/M programming is done in assembler. Unlike AMSDOS, though, lots of other languages are also available. These include C, Pascal, and Fortran. There are even a few different implementations of BASIC, although they are all somewhat different to the Locomotive BASIC we all know and love.

## **If it quacks like a duck...**

From the user's point of view, CP/M on one machine is the same as CP/M on another - just like one PC is the same as another, even if they're made by different companies. The fact that each computer manufacturer needed to customize the OS to suit their hardware, which modern PC manufacturers don't need to do, isn't really that relevant.

Even application programmers don't need to worry about the differences between different machines' CP/Ms, so a program written for one CP/M machine will work on any other. (That said, using extra features provided in later versions of CP/M can make software incompatible with early machines, just as some 6128 BASIC programs won't run on a 464.)

This makes a huge library of software available, much of which was written by people who don't even know that our CPCs exist. This includes historic applications like Wordstar, DBase and Supercalc, as well as thousands of public domain and shareware programs.

## **Where to get it**

There's a huge amount of CP/M software available online from various sources. I have gathered much of it together in one place, at <ftp://ftp.barnyard.co.uk/cpm>.

If you're not fortunate enough to be on-line but still have access to a CD-ROM drive, it is also

available from me on CD at cost price - £1.75. My address is Flat 2, 11 Beulah Road, Thornton Heath, Surrey CR7 8JH.

If you need CP/M itself, then the source code is also available on my ftp server or CD (hehe). If you have a disk drive, however, you should already have a copy of CP/M anyway.

## **About the author**

David Cantrell - [david@cantrell.org.uk](mailto:david@cantrell.org.uk) - got given his first computer, a CPC 464, for Christmas in 1984, and has played with puters ever since. He has worked for the past five years as a programmer, system administrator and security consultant. His work has included launching magazines such as FHM, Q, Empire, Max Power and New Woman online, for which he apologises, as well as working on the Playstation 2 advertising campaign, doing dodgy work for dodgy governments at a major communications and defence contractor, and writing security applications and policies for a satellite broadband ISP.

His writing and code has appeared in various open-source projects and professional publications, and his current interests include cryptography, Land Rovers, tooth-curling maths, and Roman military tactics. For some reason, the BBC saw fit to employ him recently.

# From The Archives

*How Amstrad revived a forgotten operating system*

Amongst the goodies on David's CP/M CD is a series of CP/M newsletters from the 1980s. This is an excerpt from the January 1987 issue of 'Read Only - the monthly news magazine of the Tampa Bay Kaypro User's Group and the DataCOM Super Systems(tm)'.

Dear Steve,

What prompts me to put pen to paper is the article headed 'The CPM Connection' in your Sept. '86 issue. To those of us on this side of the pond, it appears rather whimsical, since, over here, CPM is anything but dead and is in the throes of a full-scale revival!

The responsibility for this activity is entirely due to one British computer manufacturer - AMSTRAD. From launching their first machine in October 1984, they have now sold over 1 million machines worldwide. The range extends from a 64k games unit up to a 512k business machine and all have one thing in common - they all come bundled with CPM. [...]

On the PD (public domain) side, there is a lot of activity. NSWEEP attracts fans daily, MEX is far and away the most popular communication program, and DAZLSTAR is the most popular disassembler. A new program that you won't yet have seen over there is SCRIVENER, which is such a different concept that the author couldn't find anyone willing to market it, so he placed it into the public domain! You will not have seen anything like it. It's sort of a spreadsheet, sort of a text processor and has some database commands and some programming structure and ... well, I guess that it's as radical a concept as VisiCalc was when it first appeared!

This isn't all academic, by the way, because the AMSTRAD [PCW] computer is on sale in America, through the Sears-Roebuck chain. The reason that you might not have heard about it is because, over there, it's called the Amstrad Typewriter.

In England, the machine took the market by storm, because it was sold as a replacement for the office typewriter. Its bundled WP capability was stressed and the computing aspect played down. However, users were quick to realize that there was also a powerful computer with their WP and the marketing has taken this into account.

Not so in America, where it cannot compete as a computer, since its US \$799 price tag makes it a lot more expensive than the domestic competition. So, Sears sells it just as a dedicated WP system and the computing side is not even mentioned. Next time you pass a Sears store, take a look at this strange beast!

Yours truly,

Barry Pickles, 13 Norman St.,  
Manchester M12 5PR

# Twenty Questions - CPC Mastermind

For a little light relief, here are some questions to test your CPC knowledge - a small prize for the person who gets most right. Send your answers to the Charlbury address on page 1, or as an e-mail to [richard@systemeD.net](mailto:richard@systemeD.net).

- 1. What was the game 'Crafton et Xunk' called in English?
- 2. Name all the news-stand CPC magazines ever published in Britain.
- 3. From which book does the name Arnor come?
- 4. Which ROM contains the command |QWXCL?
- 5. Who was the first editor of WACCI?
- 6. ...and of Amstrad Action?
- 7. Name the CP/M word-processor most often found as version 2.66.
- 8. Which was the only game ever to receive 0% in an Amstrad Action review?
- 9. How much ROM does a standard CPC have?
- 10. What programming languages came as standard with a CPC 6128?
- 11. What was odd about the DEC\$ function on the 464?
- 12. Which team wrote the adventure 'The Pawn'?
- 13. Who ran Microstyle?
- 14. What was Richard Wilson's never-completed game creation system called?
- 15. What was Amstrad's development codename for the 6128 (allegedly)?
- 16. What does CTRL-A do in Protext?
- 17. And what does |JEUX do on a CPC Plus?
- 18. Which company manufactured 256k expansions and silicon discs?
- 19. The program 'Amsword' was an adaptation of another word-processor. Which one?
- 20. Who wrote DES?



## Keyboard Scanning

*James Hoskisson explains the CPC's rather arcane method of interpreting what you type on the keyboard*

Hi folks - it's been a while! Seeing as I'm currently averaging about one article every year I can't quite remember where I got up to with this series. I think some research may be in order...

Ah yes (he says, whilst reading his previous article). I remember now: keyboard scanning. I sincerely hope one of the other articles was about the PPI, or this article will be totally incomprehensible - not that that's anything new!

### To the point

Anyway, keyboard scanning. It's really easy to do but the way that the CPC designers implemented it makes it a pain in the posterior to carry out. Anyone who has had to scan the keyboard directly will agree with me. It certainly makes you appreciate why some games designers chose such silly combinations of keys to move the sprites around the screen!

But now I've finally managed to declare keyboard scanning to be simultaneously simple and awkward, I'd better explain how you might do it. (As an aside on keyboards, if you notice any semicolons in this article instead of apostrophes that would be because I've almost exclusively used PCs to write anything lengthy over the last few years, so I'm still not fully 're-used' to the CPC keyboard. Of course you could just put it down to bad editing... But with Richard being the consummate professional (bwah-ha-ha - Richard) I'm sure he'd have a good reason for it, should such an inexcusable thing occur!)

### Go through the PPI and take a left...

The reason that keyboard scanning is so damn inconvenient is that there is no way to get at the keyboard directly from the CPU. The design of the CPC means that you have to go through the PPI and the PSG before you can actually get at the keyboard.

Those of you who were paying attention in the last issue would know that the PSG is designed to generate sound. The rather obvious question is "Why does this chip have anything to do with keyboard scanning?". The obvious answer, of course, is "It's the CPC - since when has any of it been logical?".

The slightly more helpful answer is probably that the PSG has an I/O port on it, which is just what is needed to check the state of the keyboard. So the designers of the CPC decided to use this, rather than jack up the cost by adding more chips.

The real puzzler in this, though, is why there are so many trivial functions (such as checking the circuit board links to see what makers name should be displayed on start up etc.) connected to the much superior PPI chip, when the keyboard scanning, which is carried out much more often, is connected to the PSG, which is slower and more cumbersome to use.

My theory is that the designers of our beloved CPC fixed everything up and forgot about the keyboard; so they just stuck it on the PSG port, rather than having to mess about rearranging everything!

## Don't forget your keys

The keyboard isn't very complicated, from a hardware point of view. Basically all it is is a set of switches that return a 0 if they are depressed (no, not that sort of depressed) and a 1 if they are up. This counter-intuitive way of representing whether a key is held down or not is a consequence of the electronics, in order to save the need for extra components, and therefore saving money.

Since there are around 74 keys on the CPC keyboard (he says, after counting them) this is a lot of inputs. The way they are arranged is to split them up into 10 lines of 8 bits, where each bit represents the state of one of the keys. This is very convenient since each line can now be represented as one byte, and there are even 7 bits left over to check the state of the joystick port. Now that is convenient.

## Runb, daddy, runb

One thing that I should also mention at this juncture is the dreaded 'Keyboard Clash'. You may already have noticed this effect when you've been using the keyboard. I know I have, because I always get a 'b' on the end of a 'run' command when I type it too fast.

The problem is that when certain sets of three keys are held down, a fourth 'ghost' key appears. The 'ghost' key for any other three keys can be figured out by looking at the keyboard line allocation table.

The easiest way to figure it out is to pick two keys that are either horizontally or vertically adjacent. If you then imagine a straight line going from one of the keys to the other, you can draw two other lines at 90 degrees, which go through the two adjacent keys. These two parallel lines then tell you which keys will be 'ghosted'.

If you pick a key on one of the lines and hold it down in conjunction with the two keys that you originally selected then the key on the adjacent line will be ghosted. This is why you will get a 'b' if you hold down 'r', 'u', 'n', and an 'r' if you hold down 'b', 'u', 'n'...

I've just realised that the keyboard clash is harder to explain than I thought. Unfortunately I think that's the best I can do!

There is, basically, no solution to the keyboard clash problem. The only thing you can do is to choose keys that won't clash when your writing your high speed pure assembler super game. I don't think it would cause a problem in most situations, since it is quite unlikely that anyone would want to hold down three keys (perhaps that's why they abandoned Streetfighter II - Richard), but I suppose it's best to err on the side of caution.

## Down to the nitty gritty

I've already mentioned that you need to go through the PSG to get at the keyboard, which entails setting up the PPI to output values to the PSG so that it will select the I/O register (register 14).

Once this is done you need to tell the PSG to read from the I/O register, which is achieved by setting bit 6 of register 7 (the PSG mixer register). Now we set PPI port A back to input mode and we're away. I won't bother going into detail on how to do all this, since it was explained in the last couple of articles. I know Richard wanted articles but I don't think he would appreciate a treatise on the CPC hardware!

Right, so we've got the PSG I/O port and port A of the PPI set up in input mode. Now all we need to do is select which keys we want to check. This is easily achieved (fortunately) using port C of the PPI. This port has 4 bits set aside for selecting which keyboard line you want to select. This allows a maximum of 16 different lines to be selected, but only the first 10 numbers will actually do anything. All the other lines will return &FF if you check the PSG I/O port, so it's pretty pointless selecting them.

The keys are laid out as shown in the table on this page. You can see that the number of the line that you select alters the keys that output to the PSG I/O port. The bit number gives the state of the respective key, so if you selected line 7, for example, the byte returned from the PSG I/O port

would give the state of the X, C, D, S, W, E, 3, and 4 keys.

The directions on line 9 are for the first joystick ('Joy 0'). All of the Joy 1 inputs are mapped over line 6, in exactly the same order. This means that line 6 has a dual function and the CPC sees a movement on the second joystick as being identical to a key press.

'Spare' refers to the pin on the joystick port that doesn't do anything, in case you were wondering. It was used for the middle button of the AMX mouse, as I recall, but I don't know of anything else that used it.

That's basically all there is to accessing the keyboard. Once you've got the correct byte from the PSG it's just a matter of decoding it so that you test the bit that corresponds to the key that you want. I suppose it's worth mentioning that the byte returned from the PSG I/O port is the instantaneous state of the respective keys on the keyboard. There's none of this fancy interrupt generation or anything on the CPC! It's just a matter of polling the keyboard at reasonably short intervals. The CPC hardware may be slow by today's standards, but I don't think anybody can type faster than 1.5 million characters per minute!

We've got to the stage where I do a bit of an example. It's off the top of my head I'm afraid because I'm currently assemblerless, so if you spot anything that doesn't look quite right it's probably because it's not!

All that remains is to tell you about the Centronics Port Latch, since that will have to go in somewhere.

## Printing stuff

The Centronics Port latch is incredibly simple. All you have to do is figure out the byte you want to send to the printer and then OUT it to &EF00. This byte is then output to the printer, except for bit 7 being inverted on the way.

The byte you send is split up into the data for the printer and the strobe line. The strobe line is bit 7 with bits 0-6 being the data you want to send to the printer. The CPC only has a 7 bit printer port, so the high order data bit is always set to 0.

The strobe line is just used to tell the printer that you are sending it data. It should be set high when you are sending data in the other 7 bits of the byte.

The only other line that is concerned with the printer is the busy line, which is mapped to PPI port B, and should be checked before you try to send anything. If the printer is busy when you try to send something then it will probably just ignore it. I should probably mention that the busy line is high when the printer is busy and low if it isn't, so a simple BIT instruction can be used to determine this.

See, wasn't that simple? That just about wraps up everything I wanted to say. I currently can't think of anything to write for another issue, but you never know...

## The CPC keyboard grid

Keyboard line	7	6	5	4	3	2	1	0
0	f.	ENTER	f3	f6	f9	Down	Right	Up
1	f0	f2	f1	f5	f8	f7	COPY	Left
2	CTRL	\	SHIFT	f4	]	RETURN	[	CLR
3	.	/	:	;	P	@	-	^
4	,	M	K	L	I	O	9	0
5	SPACE	N	J	H	Y	U	7	8
6	V	B	F	G	T	R	5	6
7	X	C	D	S	W	E	3	4
8	Z	CAPS	A	TAB	Q	ESC	2	1
9	DEL	Spare	Fire 1	Fire 2	Right	Left	Down	Up

Row 9, bits 6-0: joystick controls

```

; Keyboard Scanning Routine

;; Initialise PSG and PPI

DI
; Disable the interrupts so the firmware won't butt in.
LD BC,&F40E      ;Select PSG register 14
OUT (C),C       ;(assuming PPI port A is in output mode to start with!)
LD BC,&F6C0      ;Tell PSG that we're trying to get it to select a register.
OUT (C),C
XOR A           ;Tell the PSG to go to sleep once it's done that.
OUT (C),A
LD BC,&F792      ;Set PPI port A to output.
OUT (C),C

;; Get keyboard line from PSG I/O port.
LD BC,&F645      ;Tell PSG that we want to read from its port, while
OUT (C),C       ;simultaneously selecting keyboard line 5.
LD B,&F4         ;Read I/O port value in from PSG.

.loop
IN A,(C)        ;Stick the byte we read from the keyboard in A.
RLCA           ;If bit 7 is set then carry will be set.
JR C, loop     ;Keep looping until SPACE has been pressed.

LD BC,&F782      ;Now return the PPI to the way we found it.
OUT (C),C
LD BC,&F600      ;Tell PSG to go to sleep.
OUT (C),C
RET            ;Return to BASIC and hope for the best!

```

Okay, so that program is breathtakingly unimaginative in that it doesn't do anything until you press SPACE... and then it doesn't do anything. But you get the gist. If you're lucky it may even work directly from BASIC - now that would be fun, wouldn't it? - James



## Meet the Relatives

*Brian Watson recalls some of the CPC's youthful offspring, embarrassing uncles, litigious ex-wives and other hangers-on*

### **Mea culpa (Virgil)**

If there were only 25 hours in the day and 8 days in each week, our Editor would have had this article over a month ago (I'm trying to persuade myself). Unfortunately there are not and, what with Christmas and all,...

But you don't want to hear about my trials and tribulations, you want some hot CPC action. OK, here we go.

### **Let there be cash**

When Amstrad decided that the exciting cut and thrust of home audio set-ups was insufficient challenge for their technical expertise and corporate expansionism, they looked at what was then the relatively new market for home computers and, verily, they saw that it was good - or, at least, they could probably make a lot of money out of it.

So they made the computer that was released as the Amstrad CPC 464, and then followed it up after a decent interval with a CPC 664, and then (shortly afterwards, having been criticised for introducing that only into the American market and therefore, allegedly, "keeping it from British customers") the CPC 6128.

What each of the latter two models had that the 464 did not have was "more". Or in the terminology of business, they had "value added" to a previous release. Each new computer was not just different from what had gone before, it was actually building on what had gone before to create a wish in the existing owners for an upgrade to the new model - but with backward compatibility of files and programs.

This was very clever marketing, and not a concept invented by Amstrad. But they used it to good effect with virtually all their computer ranges - including, of course, the CPC Plus range.

Speaking of those other Amstrad computers, it means the CPC has some close relations that were also built by Amstrad. And like their equivalents in the "real" world outside computing, it is useful to understand them and be able to have communication with them at times other than just at Christmas.

So let's have a quick and rather superficial look at some of these close cousins of the CPC and see what use they might be to us (isn't that often the way we regard our own relations, or is that just me who does that?). First up is the PCW series.

### **Joycean wisdom**

No sooner was the CPC 464 off the blocks and selling well than Amstrad released what was to be their major money-spinner, the PCW series.

The first PCW out was the PCW 8256 and, with 256k of memory at its disposal, it should have been better than a CPC, shouldn't it? Well no, it wasn't actually: it was considerably duller than a

CPC, comparing them like-for-like straight out of the box.

It was supplied with a 3" disc drive and CP/M (oh joy!), and could therefore run some of the same programs as a CPC (Protext, MasterFile, and Newsweep, to name but three). But it had only a mono (actually green) screen and was designed to be, and was therefore marketed as, a dedicated word processor using the bundled Locoscript program. A glorified typewriter, if you will.

Despite this, it sold like the proverbial extremely warm buns and the series soon blossomed and spread with the release of - in turn - the PCW 8512, then a PCW 9256 and finally, in the first part of the series, the PCW 9512.

As you will have deduced, the last three digits indicate the available memory of each and with every release LocoScript was enlarged to take up more and more of it! Mail-merge, spell-checking, database functions, fancy fonts ... If only they could have done it in 128k like, erm, Protext.



## Clear as mud

As each new member of the PCW series was released, better printers were supplied as standard with the computer and the mono green screen gave way to a brilliant new ... mono white screen. Doh! What a missed opportunity!

Text is tough enough on the eyes, but the effect can be ameliorated a little (and such things as italics and bold passages can be made more legible) if the colours can be different in different parts of the screen, just like on a CPC.

The 9xxx series PCWs were fitted with a 720k disc drive which makes a CPC's communication with them more difficult than with an 8xxx series - PCWs can read in data from the smaller format discs from earlier PCWs and CPCs, but not vice versa without special software. However, cable connections are an option if the computers are adjacent.

I'll return to this, and the ease or otherwise of your CPC communicating with the rest of the Amstrad ranges that I'm covering here, in more detail in the next issue.



## **Sense at last**

Just when the world of writers and vicars was reaching saturation point with the old PCWs, and you will have noticed the same upgrade path habit had been cultivated in their PCW-using adherents, Amstrad pulled a new rabbit from the hat; the PcW9512+

As well as apparently running out of upper-case Cs at this point, Amstrad were also running out of 3" drive mechanisms. So they switched to fitting 3.5" units for the "new" PCW computers. In fact, it was not until the PcW 9256 was released a little while afterwards that the PcW case got a slim-down case redesign. Otherwise the 9512+ was very much just a rebuild of the old 8xxx series, but to be able to use the bigger disks.

The PCW/PcW range was still going well for Amstrad when suddenly, with a minimum of PR fuss, writers to the 8-bit press started reporting seeing a new PcW on the shelves of the big box-shifters like Comet and Argos. It was badged up as a PcW10, and it had the look of a 9256 but with 512k of memory.

I suppose that if what you want from your 8-bit computer is a relatively enormous storage capacity for your MasterFile database or for your text, and a word processor capability that produces very fine printed output to the page (if you don't mind waiting a while for it to transfer to the paper), a PCW/PcW makes quite a handy second Amstrad for the desktop. Communication with your CPC will not be completely straightforward, but will not be an insurmountable challenge either.

## **Designs on the future**

I must also mention here that there is a lot of dedicated software that has been written for the PCW series that most LocoScript users don't even know exists. (PCW users are more dedicated in their advocacy of their computers than any other I've known. Most of us have a sense of the limits of our CPC. Even the least technical PCW users, on the other hand, think nothing of adding laser printers and hard drives to their machines. I once had an "I know my rights" user of RoutePlanner PCW whinging vociferously that it wouldn't work with his laser printer. It would, of course, he just hadn't read the manual for the laser printer interface. - Richard)

The best known 'other' PCW program is probably MicroDesign, which branched off from its CPC cousin at a fairly early stage and can produce quite remarkable results from the PCWs' larger memory.

For yet another example, a customer of mine still uses a PCW-only program to calculate the costs for the quotations and billing in her framing business and she sees no reason to upgrade now or in the future.

"Just because the latest date it will show is 31st December 1999, that's no reason to dump a quick and easy program," she says. The same argument could equally be applied to the PcW version of RoutePlanner.

Having written a few plays, books and articles myself and edited 20 editions of a magazine that was largely text, I can't see how Amstrad got away with the PCW/ Locoscript concept for so long when the CPC series did it just as well, and faster, running either Prottext or Brunword.

## **Runt of the litter**

Just before we leave the PCW series, I must make mention of the PcW16. It is a curious (I was going to use another word, but 'curious' won't frighten the horses) hybrid of 8-bit and PC technology that runs slower than any computer has a right to do - despite the fact it has a Z80 processor five times as fast as the CPC's - and crashes more often than I used to when I was an eager young sales rep.

It has its own suite of supplied programs, none of which are derived from any of those on a CPC, and it uses PC-format 3.5" disks. Cable it up to your CPC if you like, but a file conversion program on your CPC will do a better job of reading a PcW16's data from a disk. As to a PcW16 reading in data from a CPC program, it's a beast of a job, but it can be done. More on the "how" of it next

time.



## Lap dancing

In September 1992, Amstrad launched the NC series of portable computers, a precursor to today's nifty PC and Mac laptops. Because it was Amstrad, they did things differently to what might at first appear most logical - but they did things well.

The NC100 (pictured right) was launched with 64k of RAM, a built-in word processor based on Protex, a spreadsheet program, a database address book, a clock (with alarm functions), a calculator and the Acorn BBC version of BASIC. Access to the main programs is via coloured 'hot' keys - a great idea that has since been used elsewhere. For those who want to work with a larger memory storage area, the NCs will accept PCMCIA cards. (The NC100 was very similar to Clive Sinclair's earlier Cambridge Z88 - Richard)

Apart from the immediate familiarity of its word processor, what interests the CPC user is the fact that, at the back, it has both RS232 serial and Centronics parallel ports to communicate with the outside world. The outside world includes CPCs.



## Oooh la la

In April 1993, Amstrad launched an NC150, but only in Italy and France. Like the CPC 664 in its day it was a halfway house on the way to what followed as a general release shortly after. In October of 1993, Amstrad released the NC200.

The new model had 128k of memory, a double-height screen and (praise be!) a 720k 3.5" disk drive. At last the NC could easily transfer information in and out without the need to mess about with cables, but only if your computer had a 3.5" drive too and could read the NC200's disks. (They were PC-format, though, so most could - Richard)

Still, I had one of the NC200s and found it a great little computer if you could keep up with its battery consumption. A mains electricity adapter is a must!

## Occasionally innovating

Also in 1993, Amstrad released what was probably the world's first computer that would recognise real handwriting on its screen. This was the PDA600, with 128k of RAM, and compatibility with the same PCMCIA cards that the NC series could use. A PDA700 (pictured) followed afterwards.

As this is straying into the same sort of PC-compatible territory that is occupied by Amstrad's 1512 and 1640, I'll leave that brief mention of them as it stands for now, but will cover connectivity (briefly, I promise!) with all three in the next issue.



## Germany calling

Well, that just about covers - albeit in brief - all of the immediate family of the CPC. But most families have 'black sheep' and the CPC series is no exception.

There exists a computer called a KC Compact that was made in the old East Germany by a company called VEB Mikroelektronik. It's a clone copy of the CPC that uses at its heart a Zilog Z8536 chip with a patched CPC 6128 operating system chip and a Locomotive Basic v1.1 chip. Apparently, the KC is 95% compatible with an Amstrad CPC's programs. East European knock-offs of Spectrums were two a penny, but the KC was the only known CPC clone. More on that oddity some other time, perhaps.

Shall I cover here the (formerly Sinclair) Spectrum itself, which Amstrad took over towards the end of its life? No, I think I'll save that for an article all to itself.

But let me say as I close that I am immensely indebted to Emmanuel Roussin and all those other fine fellows who compiled the [comp.sys.amstrad.8bit FAQ](#) (frequently asked questions) file for much of the information I have included in this article.

## Postscript

There's one oddity that Brian has, understandably, not mentioned: ANT, or Arnold Number Two.

At the same time as Amstrad were designing the PCW, they came up with a replacement for the (still youthful) CPC. It shared many features with the PCW, such as its larger memory and its exceptionally elegant screen-handling hardware, but boasted colour, sound and CPC compatibility in addition. The aim was to see off the Atari ST and Commodore Amiga, next-generation games machines already on the horizon.

A few of the PCW's design features which don't appear to make any sense on their own stem from this shared ancestry. But the project was dropped at a fairly early stage, and the CPC Classic lived on before eventually succumbing to the Plus.

The period around the CPC's launch wasn't an auspicious time for 8-bit computer manufacturers. Sinclair's next-generation QL ('Quantum Leap') bombed, partly due to non-existent quality control, buggy firmware, and the daft decision to use high-speed tape Microdrives rather than discs.

They then started work on a direct successor to the Spectrum, codenamed Loki (after the crafty Norse god); and a less ambitious SuperSpectrum project was proposed. Neither of them saw the light of day.

The Amiga proved a profitable buy-in for Commodore after their replacements for the C64 and entry-level Vic-20, the Plus/4 and C16 respectively, also flopped. Perhaps it was just as well Amstrad stuck with what they knew.

Or perhaps not. The first results of Amstrad buying Sinclair were the Spectrum Plus 2 and Plus 3. The former was a bit like a 464, with a good keyboard to replace Sinclair's heroic failure and a built-in tape recorder; the latter was like a 6128, with a built-in 3in disc drive. But then they launched an entry-level PC, the Sinclair PC200, to compete head-on with the ST and Amiga. As with other PCs of the time, it had 4 colours and beepy sound. Oddly enough; it tanked. Richard



## Programmers' Patch

*In a change to the advertised programme, Matthew Phillips examines exactly how the CPC stores information on a disc*

I promised that we would turn our attention to disassemblers this month, but I am afraid you are going to be disappointed (or pleased, depending on how you feel about disassemblers) as I am going to leave that for another time.

Looking through my old WACCIs, I don't remember many articles looking at discs from the technical point of view. Issues 128 and 129 (approximately March and May 1999) have a couple of articles by James Hoskisson on the chip which controls the disc drive, the FDC, but what I want to look at is how the data is actually arranged on the disc, and how we program the operating system or the FDC to get the data on and off the disc.

### History in the reworking

The Amstrad CPC and PCW series use a disc format compatible with CP/M. In the case of the CPC series, Amsdos is compatible with CP/M 2.2, whereas the PCW uses CP/M Plus (also known as CP/M 3.1). CP/M Plus has facilities for (very insecure) password protection of files, as well as date stamping which is more sophisticated than that on Microsoft Windows! You can use these facilities from CP/M Plus on a CPC as well, but you may then have problems using the discs from Amsdos.

CP/M was the first standard operating system for microcomputers, but unfortunately its disc formats were far from standardised. There are hundreds of different incompatible disc formats around. If you want to see the range, get hold of a copy of 22DISK for the PC (WACCI PD disc 42). The demo version supports about 150 formats, whereas the full version supports over 400 (and there are several more that I know of, even then). Unless you are extremely lucky, none of these formats will work in your Amstrad CPC computer, even if you were given examples on a 3-inch disc.

The reason for this incompatibility is that while CP/M specifies the logical arrangement of the data on the disc, it does not require computer manufacturers to use any set physical arrangement. Let me explain the difference.

### The word of Amstrad

An Amstrad CPC data format disc can hold 180K. In a logical sense, we can just regard the disc as being a sequence of 184320 bytes. CP/M stores information in the directory of the disc so that it knows that a file occupies bytes 2048 to 4095 inclusive, for example. Physically, however, the data is chopped up into sectors, and stored on different tracks on the disc. The standard part of the CP/M operating system, written by Digital Research, does not know what sector numbers and tracks to look for. These are specified in the part of the BIOS which is provided by the computer manufacturer.

As a result, most manufacturers had a go at providing their own incompatible formats. Even Amstrad, fairly late in the CP/M scene, managed to provide their PCW computers with a disc

format which cannot be recognised by the CPC! To be fair, there was a good reason for this, which I may go into later.

## Disc structure

The recording surface on a disc is organised into tracks. There is a read/write head in the disc drive (a bit like the head in a tape recorder) which can be moved in and out, nearer to the centre of the disc or further towards the edge. It is moved by a stepper motor a set distance each time.

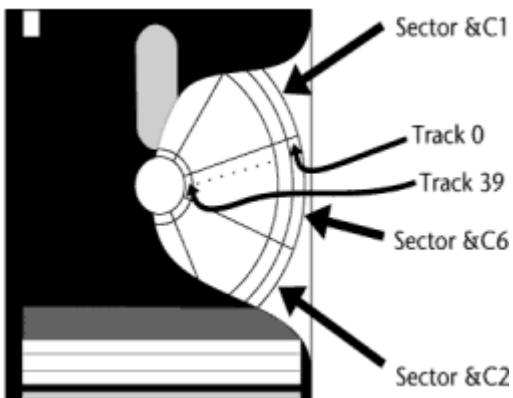
As the disc spins, the tracks form concentric circles covering the magnetic surface. In 3-inch drives used with CPCs, the head can be stepped into forty positions (actually, forty-two generally) and so there are forty tracks on a disc, numbered in true programming practice from 0 to 39.

Later disc drives, like the second drive of the PCW 8256, or any drive on a PCW 9512 or a modern PC, have a more accurate recording head which is stepped at half the distance, to give eighty tracks. They also have two heads, to allow the computer access to both sides of the disc without turning it over.

When a disc is formatted, the computer divides each track into sectors. Each sector includes a four-byte identifier, whose values are conventionally referred to as C, H, R, and N:

C - is the number of the track (also known as the cylinder number)  
H - is the number of the side of the disc (the head number)  
R - is the number of the sector  
N - is the size of the sector

The C and H values are not quite as obvious as they look. They will usually be set up as one would expect, so that the ID of each sector on track 6 of side 0 has a C value of 6 and H equal to 0. You can be cunning though, and format sectors so that the C and H values in the ID do not correspond to the physical location of the sector on the disc. The values are stored in the ID mainly to help the disc drive controller chip, because with older drives the stepper motors may not position the head accurately every time. By checking the values in the ID, the disc drive controller could see whether the motor had done its job properly, and if not, instruct it to step the head again.



## Exceptions to the rule

Funny values of C and H are mostly used in copy protected discs, as they stop simple disc copiers from working. You will also come across the situation with a 3.5" disc formatted on each side using a side switch. Although all 3.5" drives have two heads, Amstdos cannot make use of them. The side switch forces the drive to access the other side of the disc, but the computer still thinks it is the first side, so when you format side two, the H values being laid down will be zero, just like those on side one. With some clever code, it is possible to read both sides of one of these discs without switching the side switch, but on most computers you will have to instruct the disc controller chip directly to achieve this.

The number of the sector is referred to as R (I don't know what that stands for). (Record, I think. -

Richard) It can be any value from 0 to 255. The sectors on a track needn't be numbered in any particular order. In fact you can format a track so that more than one sector has exactly the same number, though this makes reading the correct sector extremely difficult.

Finally, N represents the size of the sector:

0	128 bytes
1	256 bytes
2	512 bytes
3	1024 bytes
4	2048 bytes

and so on...

Mathematically, N is seven less than the base 2 logarithm of the number of bytes in the sector. Gosh, didn't you just want to know that! On the CPC and PCW, 512 byte sectors are usually used. As you can see, the smallest possible sector is of 128 bytes. This will become more significant when we look into how CP/M organises the data on the disc.

## CPC disc formats

Getting down to specifics, what can we say about the formats commonly in use on the Amstrad? Firstly, Amsdos (and Parados) rely on the lowest sector number on the track to identify the physical disc format. From the lowest sector number you can then work out the number of tracks and sides you are expecting, as well as logical CP/M parameters such as the number of directory entries and the size of a block. See the table for all the details. Some of this information is taken from the Parados manual, with the information on Ultraform corrected.

The "Extdisc formats" shown are those supported by a CP/M+ patch called (wait for it) "Extdisc" which is available on WACCI PD Disc 7. It was written by a New Zealander, and the Nigdos referred to in two of the formats was apparently a local replacement for Amsdos.

Sadly, there is some duplication of formats. Ultraform and Parados 41 are identical apart from the sector numbers, as are S-DOS, Romdos D40 and Parados 80. Romdos D10 and Extdisc Large formats are also structurally equivalent. (If I remember rightly, S-DOS was written to use exactly the same disc format as Romdos D80, so that files might be easily interchanged between the two. One day, I'll dig out the source and write an article about how it worked. - Richard)

## In practice

In a future article we will look at how you go about working out what format a disc is from the IDs of the sectors, and also how to deal with the clashes you will observe in the table, where two different formats share the same lowest sector number. But to conclude this issue's article, we will do a little programming, and see how to read and write sectors. The assembly listing this month defines two RSX commands to make this easy, and the BASIC listing shows how to use them to read sector &C1 from the first track of a data disc in drive A.

The commands take parameters as follows:

```
|RSECT,drive,track,sector,address[,bank]
- reads a sector
```

```
|WRSECT,drive,track,sector,address[,bank]
- writes a sector
```

drive 0 for drive A, 1 for drive B.

track 0 to 39 for forty-track discs, 0 to 79 for eighty-track discs.

sector the actual number of the sector, e.g. &C1

address where to put the data in memory (or where to get it from when writing)

bank 0 for the main 64K of RAM, 1 for the second 64K, etc.

I would advise experimenting with reading sectors to start with, and make sure you write protect any discs you are not wanting to wreck!

The code as it stands has several failings. For one thing, none of the parameter values is checked to see that it is in range, so you must get the parameters right or you might do your disc drive damage. Secondly, no error value is returned, so you cannot tell whether the read or write has been successful.

## How it works

Let us have a quick look through the code to see how it works. On loading the code, you must call the initialisation routine to register the RSXs with the operating system. The initialisation also looks up two special RSXs, which yield the addresses of the disc ROM routines for reading and writing sectors. These are stored in faraddr1 and faraddr2.

(Actually, we do not need to be this long-winded: as far as I know the routines are in the same place in the disc ROM whether you are using a 464, 664, 6128, one of the Plus range, or indeed Parados. The BIOS READ SECTOR routine is at &C03C in ROM 7, and BIOS WRITE SECTOR is at &C03F.)

The RSECT and WRSECT routines are very simple. First the params subroutine is called to load the parameters in from the block pointed to by IX. This subroutine looks to see whether a bank value has been given, and converts the address if necessary. The BIOS routines are called using RST 24, which takes the address of the far address from the two following bytes. See the official firmware guide, pages 19.7 and 19.8 for details of these calls (or page 62 of the alternative firmware guide).

Next time, we will start to look at how CP/M and Amsdos organise your files on the disc itself. To explore this, you will need to have access to a sector editor. There are lots of these available. I used to use one by Martin Schroeder, published in Amstrad Action, November 1987. A listing for another was published in CWTA in August 1985. Others are available on WACCI PD discs 7 and 55. (For anyone with a secret stash of PD, I would strongly recommend the Dutch program DMon - fast, clean, easy to use, and makes a decent stab at protected discs. Of course, Xexor and later French versions of Discology are more powerful still. - Richard) I have also written one myself, which is almost fit for release.

## Save your fingers

This raises an important question. In the past, we had a Programmers' Patch disc, which is WACCI PD Disc 98. This was updated every month to include the example programs from the series. WACCI PD 98 is now full, so it might be time to start a second Programmers' Patch disc.

What do you all think about that? Is there any demand for a disc, or would you prefer me just to load the files onto my web-site? If I put them on a web-site, would you like them on a DSK image, or in a Zip file? If you have any preferences, please write in to Fair Comment and make Philip's day!

As always, feedback is welcome on what you would like me to cover in this series. Just write in to Fair Comment or e-mail [progpach@sinenomine.freemove.co.uk](mailto:progpach@sinenomine.freemove.co.uk).

## The BASIC listing

```
10 MODE 2:IF PEEK(&8500)=1 GOTO 40
20 MEMORY &84FF:LOAD"sect.bin",&8500
30 CALL &8500
40 |RSECT,0,0,&C1,&9000
50 FOR x=0 TO 31
60 FOR y=0 TO 15: PRINT HEX$(PEEK(
  &9000+x*16+y),2);" ";:NEXT
70 PRINT": ";
```

```

80 FOR y=0 TO 15:PRINT CHR$(1)+
  CHR$(PEEK(&9000+x*16+y));:NEXT
90 PRINT:NEXT

```

This assumes that the machine code has previously been assembled to a file called SECT.BIN.

```

;Disc sector read/write
ORG 8500h

; initialise RSXs
LD BC,jumps
LD HL,worksp
CALL 0BCD1h

; find disc sector reading addresses
LD IX,faraddr1
LD HL,readst
CALL 0BCD4h
LD (IX+0),L
LD (IX+1),H
LD (IX+2),C
LD HL,writest
CALL 0BCD4h
LD (IX+3),L
LD (IX+4),H
LD (IX+5),C

; end of initialisation
RET

; RSX command parameters
; |RSECT,drive,track,sector,addr[,bank]
; |WRSECT,drive,track,sector,addr[,bank]
; sector is the actual sector no., eg &C1
; bank is 0-8. 0 for internal memory,
;           1 for extra 64K, etc.

.jumps
DEFW names
JP rsect
JP wrsect

.worksp
DEFS 4

.names
DEFB 'RSEC','T'+128
DEFB 'WRSEC','T'+128
DEFB 0

; general routines

.bankswitch
PUSH BC
LD B,7Fh
OUT (C),A
POP BC
RET

.convert
LD A,(IX+0) ;bank number
LD H,(IX+3)
LD L,(IX+2)

```

```
INC IX
INC IX
ADD A,A
JR Z,convjp
OR 31h
PUSH HL
SLA H
RLA
SLA H
RLA
POP HL
RES 7,H
SET 6,H
RET
.convjp
LD A,192
RET
```

```
; disc access routines
```

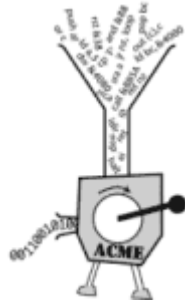
```
.rsect
CALL params
RST 24
DEFW faraddr1
JR finish
```

```
.wrsect
CALL params
RST 24
DEFW faraddr2
finish: LD A,192
JP bankswitch
```

```
; for wrsect and rsect
```

```
.params
CP 5
JR C,pjpl
CALL convert
CALL bankswitch
JR jumpparam1
.pjpl
LD H,(IX+1)
LD L,(IX+0)
.jumpparam1
LD E,(IX+6) ; drive number
LD D,(IX+4) ; track number
LD C,(IX+2) ; sector number
RET
```

```
.faraddr1
DEFS 3
.faraddr2
DEFS 3
.readst
DEFB 84h
.writest
DEFB 85h
```



# Genesis: CPC Coding from First Principles

*Richard Fairhurst begins a new series on designing and writing a major CPC program*

## Part 1: Thinking Aloud

I've got a project, and it's not a very sensible one. You may have read, here and there, about Mark Rison's brainchild, CPC/IP. CPC/IP is a whizzy piece of operating system software that gives your CPC the ability to access the Internet.

It doesn't actually access the Internet itself, though. You'd need specially-written e-mail, newsreading, or web browsing software before you could do anything useful with it. And useful isn't quite the word, because no-one, frankly, is going to choose the CPC for all their Internet needs.

But it's great fun. I think CPC/IP is a wonderfully hare-brained, futile, splendid project, and I'd like all Mark's effort to have been worthwhile. There's little point writing an e-mail program ("Ooh look, it's sent an e-mail. Isn't that exciting. I think I'll go and watch the football"), which leaves a newsreader or web browser. I've always enjoyed writing huge, over-ambitious application programs, and did so with RoutePlanner and PowerPage 64. Therefore I've decided to write a browser.

## You cannot be series

Over the next few issues of WACCI, I'll be chronicling my attempts to do so. I'm not promising I'll succeed. I'm not promising I won't get frustrated halfway through, and start work on something else (BTL, perhaps). But hopefully the program will feed off the series, and vice versa. Some of it will be a bit techy: some of it won't: but I hope you'll all be able to follow the thrust of things.

So where do you start? Each programmer has their own approach. Some plunge headlong into the difficult bit of the program, keeping all their plans in their head, just to prove the principle. Some write a 30-page exposition listing objectives, challenges, opportunities and the like. Some sketch the program out on a flowchart.

I do two things. Firstly, I work out the user interface - how the program will interact with the person using it. I conjure up the screen layout, what keypresses you'll use, whether there'll be a windows-and-pointer system, what the menus will say. This is because software should be designed with the user in mind, rather than doing it the other way round, expecting the user to fit in with the whims of the designer. (It's also because I enjoy this sort of stuff.)

Secondly, I work out the data structures - how the computer will store all the relevant information in memory. In RoutePlanner, for example, I had to work out how to store details of 2000 placenames, their location in the country, and all their road connections. Typical information would be:

```
Name:                UPPINGHAM
Settlement:          Town
Grid reference:      SK301296
Roads:
A6003 to Oakham, 6 miles
A6003 to Corby, 8 miles
A47 to Leicester, 20 miles
```

A47 to Peterborough, 23 miles

There were going to be 2000 of these. I decided to store them in order from south to north, just to make drawing the map easier. (If you want to see the map of roads around Birmingham, for example, it saves time if the program doesn't have to look through all 2000 records to find out details of nearby places, and can just concentrate on those in a certain range.)

From then on, it was all fairly straightforward. The challenge was to squeeze the maximum amount of data into the smallest space possible, so that most of Britain could be held in the 6128's extra 64k memory.

The grid references were converted to simple numeric co-ordinates, measured in eighths of a mile. As a result, each position could be stored in just four bytes.

The town names would only contain the letters from A to Z, apostrophes, hyphens, and spaces. That makes 29 characters. You can fit that into six bits - representing a number from 0 to 31 - whereas a character is usually stored in one byte, or eight bits. Bingo! A 25% saving.

I wanted to fit the road number into two bytes, or 16 bits. You can store a number from 0 to 8191 in 13 bits, and by one of those happy coincidences, you don't tend to get many road numbers above 8191 (because 8 and 9 numbers are in Scotland, and there are so few main roads in the Highlands that you rarely need to venture outside three digit numberings). So that left three bits in which I could record the road type: motorway, dual carriageway A-road, single-carriageway A-road, and so on.

Each place had a record number: 0 was Land's End, right up to 2000 or thereabouts for John O'Groats. So I used this as the destination for each road. And the distance was just stored as a number in miles. Total: five bytes for each road.

That's how RoutePlanner worked. Let's have a go at applying the same principles to the web browser.

## **The user experience**

This is all about how the user will relate to the program. A big part of this is the name. Our web browser is going to be called Heloise: it has mythical connotations fitting for such a blue-sky project, it has affinity with 'helium' - hence floating through air - hence communications 'through the ether'. It's romantic and totally impractical, because it's a romantic and impractical project. (I'm going to get into Pseud's Corner at this rate.) And I like the name.

Much of the user interface comes with the territory. It's a web browser: so the essential component is a page which you can scroll up and down. It could look like Internet Explorer or Netscape, where you have buttons at the top for the essential tools, and a scrollbar on the right to move up and down through the page. But these are designed for computers with mice. Most CPC users don't have one, making using a pointer a bit of a pain in the neck.

There's also the compelling technical factor that CPCs can scroll the whole screen faster than just part of it. So by not including a button bar at the top, we're making our whole program move faster and feel much more responsive. This is good.

Instead, all the control will be by the keyboard. Mostly, it will be using the cursor keys: up and down to move up and down the page, left and right to choose the underlined hyperlinks which will take you to another page, and ENTER (or COPY, for 464 and 664 users) to select one of these links.

ESCAPE usually gets you out of what you're currently doing, so I'm going to use that when you want to stop looking at the current set of pages and enter a new web address from somewhere else on the Internet. And the main keys will be used, sensibly enough, when you have to type text into a web page - in a search engine, for example.

## **What will you see?**

There is no way, sadly, that we are going to be able to present a full web-browsing experience on a CPC.

For starters, it'll have to be MODE 2 - which means black and white. MODE 1 text is just too fat for your average web page. And most significantly, there's no chance of graphics.

This is a real limitation and will severely limit the 'authenticity' of the browser. I wish it wasn't necessary. But just to decode GIFs - the standard file format used for illustrations on the web - takes up huge chunks of memory and processor time, requires some complicated coding, and poses problems of how you show a typically 256-colour image on a 2-colour screen. JPEGs, the file format used for photographs, are even more complex and effectively outside the reach of a CPC.

Saying that the site will be text-only makes the coding much easier. In particular, our web pages can be just like Protex documents - text starts at the top, flows down the page in one column, and finishes at the bottom. You don't have to worry about flowing it around pictures on the right, inserting little graphics for bullet points, or the like.

Further to this, all the text will be one size - the standard CPC font. We can have bold and italic text fairly easily. But there won't be any 'tables', the useful but complex web tool typically used for multi-column layouts.

Plenty of limitations, as you can see. Nonetheless, all the words, and the links, will be reproduced as they should be. It'll still be a web browser, just a rather simple one, more akin to the very first ones produced for PCs.

## Structured thinking

That's the user interface. Next, we need to work out the data structure.

There's only one significant amount of data we need to deal with: the web page text itself. As you may or may not know, web pages are written in a language called HTML (for Hyper-Text Markup Language, but you don't need to know that). This is just like a word-processor document, except that special instructions - like bold and italics - are enclosed in angle brackets. So <B> is bold, <I> is italics, and so on. These instructions are called 'tags'.

There are two wrinkles to this. One is that paragraph and line breaks are also communicated by using tags. <P> starts a new paragraph; <BR> puts a line break within the current paragraph. If you write:

```
This is the CPC's  
Brand  
New  
web browser...  
And ain't it grand?
```

in an HTML document, but don't include any <P> or <BR> TAGS, a browser will display it as:

```
This is the CPC's Brand New web browser... and ain't it grand?
```

You get the idea.

The other wrinkle is that you have 'on' and 'off' tags. One turns the effect on; another turns it off. You can tell an 'off' tag by the forward slash (/) at the start of it. To enclose a word in bold, we'd do something like this:

```
This is the <B>smashing</B> CPC web browser.
```

Essentially, HTML is a very simple language. And a CPC can easily work out that, if it encounters <B>, all the text from now on should be displayed in bold. So perhaps we don't need a data structure. Perhaps we can just load the HTML document into memory verbatim - the same way that you load a Protex document into memory, except that we load it from the Internet (via CPC/IP) rather from disc.

## Half a block

But we're not going to do that.

Two reasons. One is that HTML is simple, but it's not always efficient. For example, you can put text in a different font. One way to do this is by writing

```
<FONT FACE="arial, helvetica, geneva, sans serif">
```

Fine. But some automated web-page editing programs do this at the start of every paragraph. This is a really good way to make the document stupidly big, and hence fill up the CPC's memory after about two lines. We need to filter these out.

The second reason is more important. Any web browser allows you to scroll up and down the page at will, rather than just reading it from top to bottom. This is crucial. There might be a hyperlink halfway down the page that takes you to another, related site when you click on it - but you don't want to go there until you've read down to the bottom of the current page. So you need to scroll back up afterwards.

As we've seen, the start and end of a paragraph are marked by <P> (or, if you like, you can start it with <P> and end it with </P>). If we're scrolling down the page, this doesn't present any problems. When we come to a new paragraph, we start a new line, and print the first 80 characters of the paragraph. Next time the user presses the down cursor key, we print the next 80 characters - and so on until the end. The final line will contain whatever's left, which might not be 80 characters.

It doesn't work like that if we're scrolling up, because we don't know how many characters are in the last line. (If you don't see what I mean, look at the previous paragraph on this page. All the lines are the same length, except the last one.) If we assume it's 80, that will leave us an odd number of characters when we scroll back up to the top line of the paragraph.

## The parser

It would be loads easier if HTML text was organised into 80-character lines, rather than paragraphs. So let's do it like that.

What we need is a 'parser'. This is a cunning bit of programming that reads in the HTML page from CPC/IP; and stores it in memory how we want it. It'll read <B>, for example, and store it in memory as our special code for 'turn bold on'. It'll read in 80 characters from a paragraph, and then insert our special code to say 'the line finishes here'.

In effect, this divides our web browser into two distinct sections.

1. The parser. Reads in the web page from CPC/IP, and records it in memory how we want it. When there's no more page to read, we then move onto:

2. The page display routine - the real browsing bit. This puts the page on the screen, gets your keypresses, and reacts to them.

## There was a time

It makes sense to write part 2 first (no, really, it does). This is partly because the parser's role is to feed the page display routine, not the other way around. So by writing the page display routine first, we get a clear idea of what we want the parser to produce.

It's also because we can test the page display routine without the parser. We just dummy up a web page in our special format - ready-parsed, if you like - and see what the page display routine makes of it. Testing the parser without the page display routine is a bit more esoteric. You could look in memory to see if it's stored the page correctly, but that would be very boring.

Next issue, we'll discuss exactly how our data structure is going to function (which I've worked out already), and start writing the page display routine (which I haven't).

# Famous last words (rpt. squared)

This issue of WACCI was finished in the first week of February, so to keep to a two-monthly cycle, it'd be smashing to receive contributions for WACCI 139 by mid-March.

Once again, suggested topics might include:

- Tips and tricks for that 'old favourite' piece of software you use... or a critical reappraisal
- Details of the whizzy new software you're developing
- Why you still use a CPC
- Reviews of new software (and there's plenty of it, so if you'd like to be put on our list of reviewers, drop Richard a line. I'd especially like to hear from people with CPC Plus machines.)
- Your opinion of WACCI or the website

Needless to say, Philip would also like to receive some letters for Fair Comment, otherwise Brian will have to make them all up again.