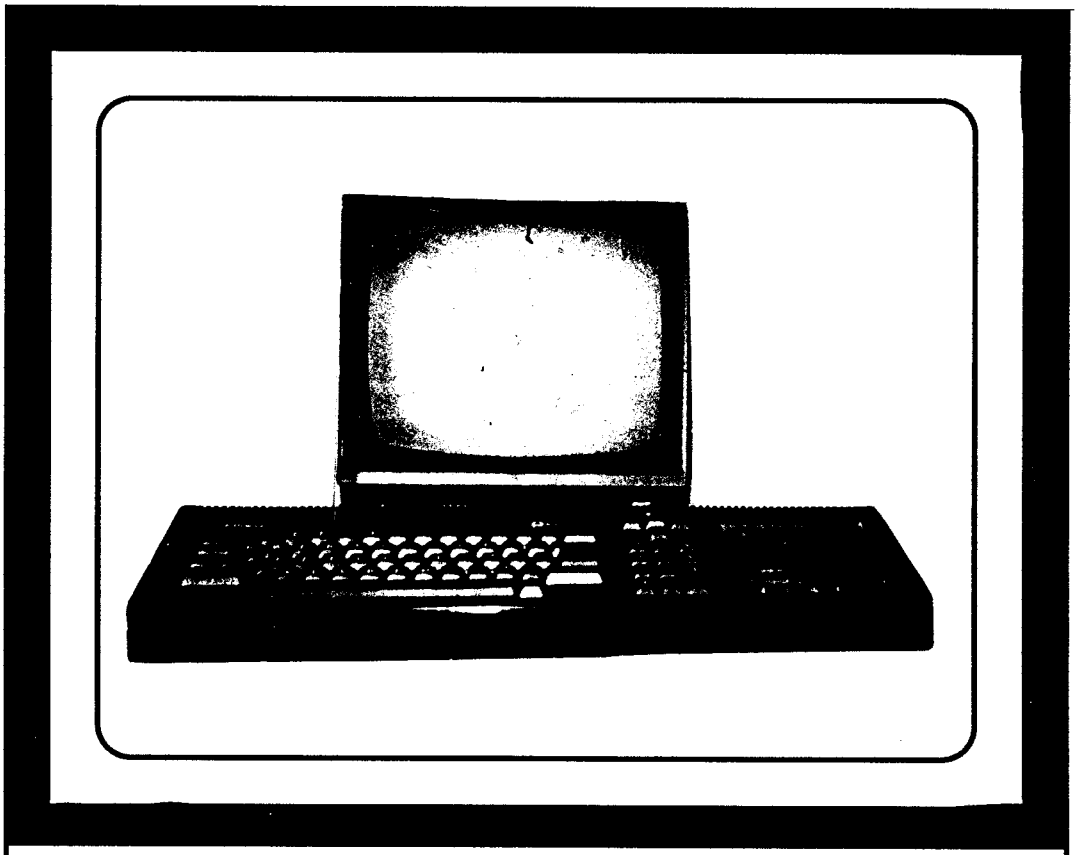
Price £1.10

PRINT-OUT

ISSUE NINE



Written by Thomas Defoe and Mark Gearing
Technical Contributor: Bob Taylor

INCLUDING:

WORDPROCESSORS
SCREEN DUMPS
LETTERS
BASIC
CPM

INDEX — ISSUE NINE

Miscellaneous

Features

Reviews

Programming

Once again we would like to express our thanks to Mr Gearing and Black Horse Agencies Januarys Consultant Surveyors for their continued support of Print-Out and for the use of their photocopier in the production of this issue.

EDITORIAL

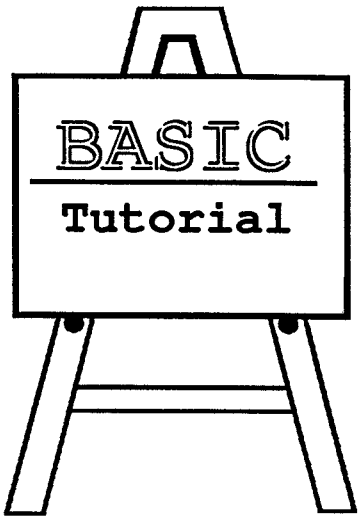
All I ever seem to do in this editorial is apologise for the lateness of the current issue of the magazine. However we'll be trying to make up for the delays over the next few months.

Since Issue Eight there have been a number of changes to Print-Out and the greatest is that we no longer fasten the magazine with the black plastic binders. One reason for this is that a few readers complained that the binder was tearing through the envelope and so copies of Print-Out were getting lost or damaged in the post. The other reason is more financial (the binders have more than doubled in price recently) and instead of charging more for the magazine we decided to get rid of the binders.

Over recent months I have been alarmed by the number of complaints that I've received about certain mail order companies; apparently people are still waiting to receive their orders after six months! On behalf of these people, we are trying to see if anything can be done, and so we are anxious to hear from anyone else who is having problems with any mail order company.

Many thanks to all of those people who have helped with this issue of Print-Out. As an incentive, the author of any article that we publish in future issues will receive a free copy of Print-Out. If you don't feel like writing something, then why not introduce your friends to the magazine. You'll receive a free issue for every new subscriber (for three or six issues) that you introduce. Just ask your friend to mention you in their letter and we'll do the rest. If you already have a subscription, we'll extend your subscription by one issue. There is no limit to the number of free copies that you can receive, so you could end up never having to pay another penny!

PRINT-OUT, 8 Maze Green Road,
Bishop's Stortford, Herts CM23 2PJ.



Beginner's BASIC

Following on from last issue's introduction to using data in structured BASIC programs, in this article we will be looking at some commands which can be used in association with DATA statements - namely DIM and ERASE.
First though, we've had several queries regarding errors in DATA statements and as they're rarely obvious, we have devoted some time to look at the various errors which can crop up. Data errors are notoriously hard to put right as the error messages never tell you the right line in which the problem has occured.

DATA EXHAUSTED

Possibly the most common error is 'DATA EXHAUSTED', and this means that there isn't enough data for the program to read. In the example below, we've asked the CPC to read eight numbers and then to print them. However, we have only given it seven numbers in Line 50, and so the CPC will still be expecting one more bit of information. When it cannot find the piece of data, it will tell us that all the data has run out.

```
10 FOR i=1 TO 8
20 READ number
30 PRINT number
40 NEXT i
50 DATA 6,7,43,12,8,14,102
```

If you come across this error, the first thing to do is to check that the FOR...NEXT loop is telling the Amstrad to read the correct number of bits of data.

The 'Data exhausted' message will be followed by a line number, and this line contains the READ command. The FOR..TO... instruction should come shortly before this READ instruction.

It is important not to change the value in the FOR....NEXT loop, unless it is incorrect, as while you may get rid of this error, sooner or later another error will crop up - and this can be even harder to sort out.

If the FOR...NEXT loop value is correct, then somewhere you must have missed out some data. The only remedy is to check through all of the DATA. This is very difficult (and boring!), especially in a long program.

As a quick check, almost all magazine listings, and most particularly machine code pokers, have DATA statements of the same length and so you should easily be able to spot which line is missing some data. When typing in any program, a hint to help avoid omitting an entire line of data, is to use the AUTO command.

Errors in DATA statements can be caused by accidentally typing a 'full stop' instead of a 'comma', especially when entering numerical data, and this is worth bearing in mind.

TYPE MISMATCH

This is another very common error, and it usually crops up when you are using hexadecimal data. As with most problems, prevention is better (and easier!) than cure. In this case, it almost always occurs due to incorrect entry of data.

The program below is typical of a machine code poker (this is the most common use of hexadecimal data). Most of the program should be fairly self-explanatory. The 'Type Mismatch' error will point to the rather curious Line 30 and it's here that the program converts the hexadecimal information into decimal.

```
10 FOR addr=&8000 TO &8005
20 READ dat$
30 byte=VAL("&"+dat$)
40 POKE addr,byte
50 NEXT addr
60 CALL &8000
70 DATA 3e,50,cd,5a,bb,c9
```

Hexadecimal numbers can consist of the numbers 0 to 9, and the letters A to F (NB it does not matter which case you use). Most Type Mismatch errors are caused by entering an illegal character in the data. The most common are using a capital letter 'O' instead of a zero; or using a small letter 'L' instead of a one.

In a program without too much data, you may just have to check all of the pieces of data - you can find out what the wrong piece of data is by using PRINT dat$

If the program has a large amount of code, it will probably have a better system of reading & poking the data. This will check to make sure that each number does not contain any illegal characters, that each line has the correct numbers in it and that no lines are missed out.

IMPROPER ARGUMENT

This error can crop up anywhere in the program and rarely seems to be related to data (in fact it doesn't have to be!). It is caused by some 'illegal' figures being used in a command.

An example of this is with the instruction LOCATE x,y. Here, if either of the variables 'x' or 'y' has an impossible value (for example 0) then it will result in an 'Improper argument'.

To sort out this problem, list the line in which the error occurred and print the values of all the variables in it. Now find any variables which have illegal values, and search the data lines where these variables were set up. In general, data related errors can be avoided by taking care when entering a listing from a magazine.

SYNTAX ERROR

One other error, which is pretty rare, is 'Syntax Error'. This is only tricky because it should not really be a 'Syntax Error'. It occurs when the CPC expects to be reading a piece of numeric data and instead finds a bit of string data, or vice versa. Again, the only method of sorting this out is to look through all of the data and try to locate the error.

DIM AND ERASE

As I mentioned at the beginning of this article, there are two commands which are often used in conjunction with data - DIM and ERASE - these two instructions both deal with arrays in the CPC.

In computer terminology an array 'is the name given to a set of data which is arranged so that any individual piece of information can be accessed by using an identifier or key'. Very interesting, but in practical terms what is an array?

An array is best illustrated by means of an example. Going back to one of our programs from the last issue (shown below), this routine prints out the names of the months, and includes nothing new. However, just suppose we wanted to store each of the names of the months, in order that we could use them in the program at a later stage. Obviously we have to use a different variable for the name of each month (so that we can use the names separately, if required). Unfortunately the length of such a program wouldn't be worth it. However the next program stores all of the data in a single array.

```
10 FOR i=1 TO 12
20 READ month$
30 PRINT month$
40 NEXT i
50 DATA January,February,March
60 DATA April,May,June
70 DATA July,August,September
80 DATA October,November,December
```

The program is not much longer than the previous one and illustrates the use of an array. In it, the variable month$ is holding the names of all twelve months. But, the number after it in brackets is the identifier (or key). This allows us to access just a single name by using a number from 0 to 11 - the range was set up in Line 10 by the DIM command.

```
10 DIM month$(11)
20 FOR i=0 TO 11
30 READ month$(i)
40 PRINT month$(i)
50 NEXT i
60 DATA January,February,March
70 DATA April,May,June
80 DATA July,August,September
90 DATA October,November,December
```

Line 10 tells the computer how large the array has to be. The command to do this is DIM (which stands for 'DIMension array') and the syntax for using it is as follows: **DIM variable(size)** One thing to note is that we only dimension the array to be eleven 'elements' large. The reason for this is that the computer starts counting at zero so by the time that it reaches eleven, it has counted twelve numbers! Hence, in Line 20 the loop is now from zero to eleven.

When an array has been dimensioned, it's like a large box which has been given a name in order to identify it. The box is split up into the required number of smaller compartments, and each of these small compartments is identified by a number. The diagram below shows how the array 'month$' is set up in the above program.

n	0	1	2	3	4	5	6	7	8	9	10	11
month$(n)	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec

Line 30 assigns a piece of data to each part of the array and Line 40 prints out the name of the month which has just been assigned. You can check that all of the months can actually be accessed individually by typing: PRINT month$(n) where 'n' is any number from 0 to 11.

MULTI-DIMENSIONAL ARRAYS

Naturally that isn't the end of the story. Imagine you were writing a game of computer chess - you would need to keep track of where all of the pieces were on the board and continually update their positions. A chess board has 64 squares, in an 8 by 8 arrangement; so we could use a normal array with 64 entries - this would be set up using this command: DIM grid(63) - but it would make more sense with a two-dimensional array.

So just what is a two-dimensional array? There are two ways of thinking of it, and which way you use depends on what the array is going to be used for. In respect to a chessboard, imagine that you have a x-coordinate and a y-coordinate with which to identify each square on the board - therefore, your two-dimensional array will be set up in this form: DIM grid(x,y) and specifically for a chess-board: DIM grid(7,7) - because each identifier of the array goes from 0 to 7, and so giving 8*8=64 squares which can be individually accessed. Just imagine needing to have 64 different variables to hold the contents of each square; it would make a program unbelievably long and cumbersome.

(x)	0	1	2	3	4	5	6	7
0	R	Kn	B	Q	K	B	Kn	R
1	P	P	P	P	P	P	P	P
2								
3								
4								
5								
6	P	P	P	P	P	P	P	P
7	R	Kn	B	Q	K	B	Kn	R
(y)								

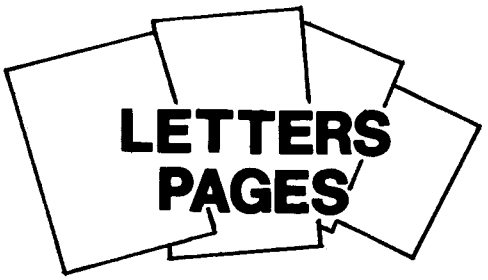
CHESSBOARD ARRAY

The second way of viewing a two-dimensional array is more helpful if you are storing a list of things. You could think of the first identifier as indicating which compartment you are looking at, and the second identifier as showing which smaller sub-division of the compartment you are accessing. To explain this, look at the table, which shows an array defined by: DIM list(comp,box) - comp=7,box=1

(comp)	0		1		2		3		4		5		6		7	
(box)	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

In this example, each of the eight compartments (comp) is further sub-divided in to two smaller boxes (box). Of course you can also have three-dimensional arrays which can be regarded using either method of representation. After that you can have arrays with as many dimensions as you like (or at least, until you run out of memory!!!). To prevent wasting valuable memory with arrays that you no longer require, the CPC provides you with the ERASE command and this deletes any arrays which you specify. Its form is: ERASE array so to delete the array 'month$' which we defined earlier in this article, you would use: ERASE month$

That's it for the moment, see you next issue with some more BASIC commands.



If you have a question, information or something to say, we would be delighted to hear from you.

Knitting on a CPC

I wonder if you know of any company that has produced a program for designing knitting or sewing patterns. The magazine, ACU, published an article called 'A Stitch in Time' in their February 1991 issue on their News page. I wrote to the Editor of ACU to ask for the address of the company who could provide me with a copy of the disc 'Designer', but did not hear anything from him!

MRS M E BROCKBANK
KENDAL

PRINT-OUT: Every time I try to do any sewing, I always manage to end up with the needle stuck in my thumb, and a fast-unravelling cotton reel rolling across the floor! So I'm afraid that I'm not going to be much help with this one. If any of our readers know of such a program, please get in touch and I am sure that Mrs Brockbank would be very grateful.

RS232 Port Leads

Let me thank you for Issue 8 of Print-Out and especially thanks for the reply to my enquiry about transferring binary files which you included on page 11 - it worked perfectly and I was able to transfer the games to the disc.

Before I close can I chip in with an enquiry which may be a little unfair to you as it's really a hardware question. I recently obtained a second hand serial interface board which has all of the necessary chips as far as I've been able to ascertain. It is not specifically for the Amstrad but I had the idea that I may be able to use it to obtain an RS232 port.

It is a question of getting the connections correct. I can trace out the data lines but I am not sure of the control connections. Ideally, if I could obtain a copy of the circuit diagram of the Amstrad serial interface that would give me the answer.

JIM PROCTOR
NEWCASTLE

PRINT-OUT: I'm glad to hear that you found the transfer advice helpful, although it will only work with unprotected binary files of course. I wrote to Amstrad about circuit diagrams and they suggested that you got in touch with CPC Ltd who supply the service manuals and some parts for the CPC computers. Their address is: CPC Ltd, 194-200 North Road, Preston, Lancashire PR1 1YP
(Phone: 0772 555034)

Plus or Minus...?

To say that I'm disappointed with the 6128 Plus is quite mild. I've a ROM box with Protext and Prospell which I cannot as yet use. From what I have been told, the incompatibility of the expansion port with an adaptor on is due to the Plus itself. So for the time being I am having to 'make do' with Mini-Office II.

I have managed to obtain an Expansion port adaptor from Microstyle. However, my ROM Board still did not operate with this adaptor. At the suggestion of Phil Craven of Microstyle I contacted Rombo Productions in Scotland who advised me to return my ROM board complete with the adaptor and ROMs fitted. It appears that a modification needs to be carried out on the board to suit the Plus models - this modification is going to cost me approximately £10 to £12!

When the board is returned I need to see if my Genius Mouse and its interface will operate successfully or if that will require some attention. Further to all this I did mention to Phil Craven that tapes were a 'no go' with the Plus having no tape socket. He did say that when the warranty period on my Plus had expired, he would possibly be able to modify it to accept tapes.

All other aspects of the Plus are fine, especially the graphics when playing a cartridge game.

MR D A WEBB
CHORLEY

PRINT-OUT: It seems that a number of readers are having problems with upgrading to a Plus computer - so much for the Pluses compatibility with the CPC. Still help is at hand in this very issue of Print-Out, as we have a rundown of most of the serious hardware and software that works on the Plus computers.

Alternative Firmware

I'm glad to know that someone is prepared to help CPC owners - especially in the serious field. There are far too many games about and the main magazines are swamped by them. Please don't let it happen to Print-Out.

I understand that Digital Research are thinking about publishing an updated version of the Firmware Manual - long, long overdue. Have you heard anything on this? I need it and I want it NOW, as in yesterday! I know what you are thinking - 'don't we all?'

MR T NORRELL
LARNE

PRINT-OUT: It appears that there may be some good news for CPC users. I recently received an Amstrad CPC price list from WAVE (all 29 pages of it) and whilst wading through it I came across an absolute gem - the genuine firmware manual (SOFT 968) - on sale at £17.96 which includes postage & packing. Now, I don't know how many of these WAVE has, but if you want one I suggest you give them a ring on 0229 870000 (Mondays to Fridays) and see if they can help you. Oh, don't worry, we'll never let Print-Out become a games magazine.

Down in the Dumps

Please could you or one of your readers write a proper printer dump that will work on an Epson compatible printer. I don't want ones that print the screen at double height or width or only works with Mode 2 screens, like the others that I have tried. All I want is a nice simple one that prints any screen.

ALASTAIR HENDERSON
OCKHAM

PRINT-OUT: No sooner said than done!!! Turn to page 18 and all your prayers will be answered. Our resident machine code wizard, Bob Taylor, has written a very good shaded printer screen dump that will work in any mode.

Viking Invasion!!

Hello again, this is the Norwegian Viking calling!!! I was just sitting in a chair doing an essay on Sir Isaac Newton, when I suddenly realised that I've not written to you for months.

I've been working a bit with machine code in my Easter holiday, and I really find your assembler useful. Actually I don't use the others any more. By the way I was trying to copy some game-discs yesterday but I've only Siren's Masterdisc, so it didn't work.Do you know if Siren's Discology still is available? I've been also looking for Amstrad Action in various shops, but no luck. I have only seen ACU, which I also buy once a month.

By the way, are you a relation of the man who wrote the books about Robinson Crusoe????

BJORNAR SAETERNES
NORWAY

PRINT-OUT: It's nice to hear from our foreign readers, except that it puts me to shame when people can write English so well, and I can't speak a single word of Norwegian. Still onto your questions: I believe that Siren do not produce Discology any longer. However, WAVE is an absolute goldmine for discontinued programs and hardware, and another quick look through their price list shows that they stock it at £13.43 including postage and packing (you will need to phone to find out how much it costs to send it to Norway). Their address is:

WAVE, 1 Buccleuch Street, Barrow-in-Furness, Cumbria LA14 1SR.
Phone: 0229 870000 (Mondays to Fridays)

No, I'm not related to the author, Daniel Defoe, I'm afraid - my family were originally French Huguenots who fled to Britain in about 1685. Incidentally, we've managed to trace our family tree back to 1736 - I don't suppose anyone really wanted to know that!!!

Multiface Jump

I have a problem concerning my Multiface 2. I'm trying to use the direct jump and I wish to jump to an address at &4000 in bank &C0. Using the Multiface, I've poked &00 into &2000 of the Multiface's RAM, &40 into &2001 and &52,&55,&4E into &2005-&2007 and at &4000 I have loaded a short program.

```
ORG &4000
LD A,55
CALL &BB5A
RET
```

I have tried it with and without the 'ORG' and the 'RET'. Usually the screen corrupts or nothing happens. So could you please tell me how to execute the code using my Multiface.

LEO CRAWFORD
WEST MALLING

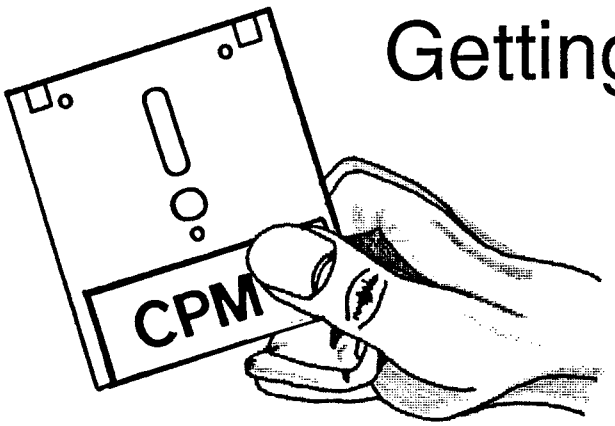
PRINT-OUT: Yes, bit of a tricky one this. It's all to do with the value that you poke into &2002-&2004 and these numbers must be exactly right. I too have had much the same problem, but have been unable to solve it. This being the case, I've passed your query on to James Verity at CPC Network (the people who make Tearaway for the Multiface) and hopefully they'll be able to come up with the answer.

Database Query

I'm looking for a database to use in conjunction with Protext. I would mainly want to store people's names and addresses and then print them out when I needed to. I have heard that Protext Office and Protext Filer from Arnor are no longer available and so I was wondering whether there was any way of doing this. I also own Promerge Plus on ROM and thought about trying to write my own database using Promerge stored commands.

JOHN WELCH
EDINBURGH

PRINT-OUT: As you say, Arnor have stopped making Office & Filer, but WAVE do stock some copies. I would think that it would be difficult to design a database using the stored commands as I cannot see any way of saving a modified data file from within the program. Arnor recommended Masterfile as being a database which will work with Protext.



Getting to Grips with CPM

Introducing the system: Part 3

As none of our readers offered to write the CPM column I've been left with the unenviable task. Over the past few weeks, I have read up on this complicated operating system, changed loads of files and corrupted so many discs that I've now lost count. In the process, I've altered my CPM disc beyond all recognition & have made it into a quite usable beast. One thing that I have learnt is that CPM can be incredibly powerful and useful.

Because both 6128 and DDI-1 owners have access to a copy of CPM 2.2, the main article will deal with CPM 2.2 specifically and then at the end I will mention a few of the special functions available to CPM Plus users.

The main reason why CPM isn't more popular is because it is so different from BASIC. There are no helpful error messages, gone are the instantly recognisable commands and instead there is just the terrifying 'A>' prompt.

If you are serious about using and programming CPM Plus, then I recommend you get hold of an excellent book called 'The CPM Plus Handbook' by Digital Research and Amstrad (ISBN 0-434-90321-3). It's a mine of information about the operating system, contains over 500 pages, is heavy-going and costs a whopping £20.

For those of you who own CPM 2.2, the book you want is 'Operating the Amstrad CPM 2.2 Guide' which is also published by Amstrad (SOFT 06016) and costs £9.95.

What is CPM ?

CPM (Control Program for Microcomputers) is an operating system which allows you to talk to the computer and so to get your CPC to do what you want. Our CPCs are supplied with another operating system called BASIC, and most people will be familiar with this means of communicating with their computer. The CPM Operating System consists of two main parts, and they are:

a) The CCP (Console Command Processor) - this allows you to actually give the computer commands, and controls the execution of programs.

b) The BDOS (Basic Disc Operating System) & BIOS (Basic Input/Output System). The BDOS and BIOS contain the code which makes up the operating system and allows the computer to interpret programs and run them on our CPCs.

As anyone who has really studied the Z80 microprocessor will know, it is very difficult to actually program the microchip itself. The closest most people come is to use machine code and the firmware calls - few people would dare to try and send data directly to the Z80. Likewise in CPM, no-one would wish to talk to the microprocessor and CPM provides various routines in the BDOS & BIOS to actually do the tricky job of talking to the Z80. As users of CPM, we can take advantage of these routines and spare ourselves from the complexities of the hardware.

The BDOS contains 'high-level' routines which can perform quite complex tasks such as opening and closing files, reading and writing data and even erasing and renaming files. The BDOS itself uses the 'low-level' routines of the BIOS, which really do the talking to your CPC - thus only the BIOS needs to be specifically coded for each different computer that CPM runs on. Amstrad have put the BIOS on the disc ROM and the BDOS is stored on the System Tracks of your CPM disc. These tracks are copied onto a disc whenever you give it a System format; thus CPM can only be booted from a System Disc.

Before we investigate CPM any further, we need to look at some of the common terms used when describing this operating system:

BDOS - contains routines to perform specific tasks to do with the disc drive; it uses the BIOS to do these routines: means Basic Disc Operating System.
BIOS - contains low level routines which tells the hardware what to do; it is on the disc ROM and is machine specific: all actions are performed using the BIOS: stands for Basic Input/Output System.
BOOT - when CPM is loaded into the computer's memory, it is said to be 'booted'.
CCP - this is the Console Command Processor: it lets you enter commands, & then interprets them, finally it informs you of the outcome.
COLD START - another name for 'booting' (loading) the CPM operating system.
DESTINATION DISC - this is the disc on which you want data to be written.
READ ONLY - this means that you are unable to write data to the disc.
READ/WRITE - means that you are able to read and write data to the disc.
SOURCE DISC - this is the disc from which you want to read data.
SYSTEM FORMAT - this refers to a disc which had been formatted in a particular way & has the CPM **SYSTEM TRACKS** written on it; these tracks are parts of the disc which contain the CCP and BDOS (essential for CPM to work).
TPA - this stands for Transient Program Area & is a section of the CPC's memory into which CPM programs may be loaded.
TRANSIENT PROGRAM - this is a program which has to be loaded into the TPA to be executed by the CPC; some of the more common routines are included in the operating system and do not have to be loaded from disc.

For CPM Plus owners, the rest of the article equally applies to this operating system. However, CPM Plus is supplied with a way of producing graphics as well text: this is called GSX (Graphics System Extension). Although it is perfectly possible to write your own routines to allow you to create an artistic masterpiece in CPM, it's very tricky and beyond the scope of this article and so you would probably need to invest in a 'dedicated applications program' to be able to make the most of these extra functions. The reason that I mention this now, is that GSX doesn't use either the BDOS or the BIOS to create these images and so could be regarded as a separate part of CPM Plus.

Next issue, we will start looking at some of the more common CPM commands and seeing what happens when we type |CPM.

PLEASURE DISC ONE

There seems to be a large number of software writers amongst our readership and the latest arrival to the homebrew pages is Kevin Heywood. Pleasure Disc 1 is a compilation - and is hopefully the first of many.

The first program that I tried was Dead-Zone. The aim of this game is to wipe out all of your opponent's forces, and to do this you need to move your men and cannons into a position where they can blast the enemy. Your men have bazookas & each side also has a couple of photon cannons - the cannons are more deadly and have a greater range. On the screen there are trees and bushes which can act as obstacles to your men and missiles.

The game is well presented and the general idea is very good (although maybe not entirely original) with experimentation needed before you work out the exact distance needed to level your opponent's cannons and men. However, the game is a bit limited as it is played on just a single screen; even so there are some nice touches like the birds fleeing the trees when one of your rockets hits them.

The greatest drawback, and this applies to several of the games on the disc, is that you can only play against a friend - my view is that if a game is played on a computer there should be an option to play against the CPC. Apart from this it is a great game to play, although I wonder with how much ultimate appeal.

The next program is Fruit-Run. Many homebrew writers seem very fond of fruit-machine simulators and so this game needed to be very special to stand out from the crowd. Whilst it was competent and reasonably fun to play, there just wasn't any hectic sound, flashing lights or compulsion for 'just-one-more-go'. Still it did have several features, such as Holds, Hi-Lo, Nudges, Fruit Spin and Cash-Box to liven it up.

Match-It was a very attractive and well presented version of the old game in which you have to remember where cards are and turn over matching pairs. I liked this game enormously and my only complaint, as I have already mentioned, is that there's no option to play against the CPC.

Killabeez was an arcade game very much in the style of that old shoot-em-up, Galaxians. Well programmed, quite smooth and colourful the program had the added twist of keeping your honey pot topped up by shooting the bees and avoiding the bombs that they dropped. To make it more interesting, there was a bonus level in which you had to avoid some 'flowers' and collect others. Although the game was fast, I did feel that perhaps it was just a little bit to easy; giving any game the right difficulty level is one of the hardest things to do.

The final game on the disc was Puckshot. Another two player game, you had to steer a little spacecraft around and fire at a ball, until you got it into your opponent's goal. This was probably the least enjoyable game on the disc, but was still quite entertaining for a while.

It was quite a mixed bag of games, but I feel that most people would enjoy at least some of the programs on the disc. Pleasure Disc 1 costs £2.50 (if you send your own tape or disc) or £4.90 including disc, or £3.30 including tape. You can order your disc from:

Kevin Heywood, 18 Sinclair Avenue, Banbury, Oxon OX16 7DW.

THE 'SNELL' COMPILATION

I last reviewed software from Barrie in Issue Six of Print-Out. I thought the programs then were very good indeed. So it was with an air of expectancy that I loaded the disc from Barrie containing all his latest software. He certainly has been busy! Barrie has written a new game called Cribbage, a Calendar program and Blackjack has been re-written to give extra options. To complete the collection for his new disc, he has included Wordsearch which remains unaltered from when I reviewed it in Issue Six of Print-Out.

Cribbage is a British card game, originally intended for two players. However there are many different playing options & some of the variations are described in the very extensive hand-out which accompanies the game. The basic aim of the game is to score points without letting your opponent score.

The most common form of Cribbage is the five-card two-player option, and this is the one Barrie concentrates on when describing how to play the game. He takes the reader through a game & his instructions, although not simple are relatively easy to understand - don't be put off by the volume of notes!

Now onto the game which is written in a clear and systematic way. Again, his instructions are easy to follow, although it is necessary to have read the hand-out before you start the game.

The game has loads of appeal and you certainly won't get bored with it if you like computer card games. If you don't, then this game is not for you, as it is complicated and not just a quick game of snap or pontoon!

There is no sound and little colour but to be honest a game of this type does not really need it. The screens are unfussy, uncluttered and do the job in hand very well. I doubt whether it remains as good fun with computer as it does with other people but that's up to you.

The next program on Barrie's disc is called Calendar and he claims it is the only accurate British calendar program for the CPC range in existence. It deals with the years from 1 AD to 3999 AD, and it automatically takes into account the changeover from the Julian to Gregorian calendar in September 1752.

All you do is type in the year you want, and the computer then prints out the calendar for that year. Whilst it is interesting to find out what day a special date fell on, I think that ultimately this program has very limited appeal. But the information on Barrie's sheet detailing the history of the calendar through the ages is a good read.

Blackjack was a very good program last time I reviewed it and so I was unsure just what improvements were needed to it. Although the gameplay is identical to before, the game flows more smoothly as there are less key-presses to hold the game up. In addition, Barrie tells me that there are some extra options - there seemed to be plenty before - and all-in-all it's still an excellent game for the gambler in all of us.

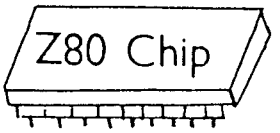
The disc is rounded off with Wordsearch, which is identical to the program we reviewed in Issue Six. Overall, the disc contains a good mix of programs, with the emphasis placed on card-players, and is well presented with the many notes & options, and represents very good value-for-money at £4.00 (if you send a disc) or £7.00 if Barrie supplies the disc. To order your disc, write to:

Barrie Snell, 19 Rochester Road, Southsea, Portsmouth, Hants PO4 9BA.

Machine ....Code....

Programming the Z80

Multiplication



Throughout the next few issues, we are going to be looking at the various ways that calculations can be performed in Machine Code. At the end, we will put some of them together to produce a simple calculator which can be used either as part of a larger machine code program, or by calling it from BASIC.

In this issue, we are going to write routines which will allow us to multiply some small unsigned numbers by each other. As with everything, there are several ways of going about this and we'll be using the simplest method in this article; this is the use of straightforward addition to mimic the more complex functions.

Times Tables

People multiply numbers together by different means but most involve using the knowledge of multiplication tables (eg 7 times 7 is 49). However, an alternative to this is the use of repetitive adding, and this is the method we are going to use when writing our multiplication routine. The Z80 microchip inside our CPC is particularly good at adding numbers together and can do many such calculations a second; thus multiplying even large numbers together will not take too long.

To multiply the numbers 7 and 3 together, we just tell the computer to add the number seven to itself three times (ie. 7+7+7 = 7x3 = 21) and that is really all there is to this form of multiplication.

The only real complication lies in making sure that the computer can store the answer correctly. If you think back to the early articles in this tutorial, you will remember that a single register can hold an 8-bit number (ie. a number with a value of between 0 and 255) and a register pair stores a 16-bit number (with a value from 0 to 65535). As long as we only multiply two 8-bit numbers together, we can store the result in a 16-bit register pair. But if we start using numbers which produce a result greater than 65535, we need to use a different method of storing the answer.

Therefore for this issue, we will concentrate on multiplications which produce answers that we can store in a standard 16-bit register pair - in future issues, we will be looking at larger sums.

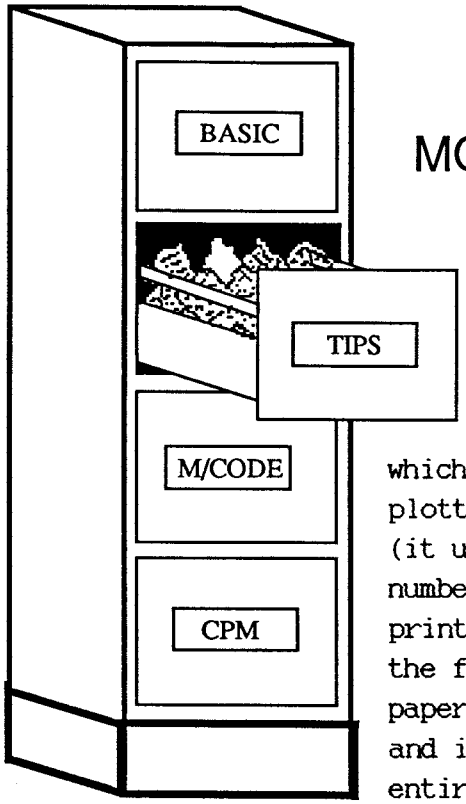
Method

What does it do? Register E holds one of the 8-bit numbers, and Register C the other. Register Pair HL is set aside to hold the result of the multiplication. A check is then made to see if E has a value of zero; if it does the routine jumps straight to the printing routine - there's no point in doing any calculations as anything times zero still equals nought at the end. If E is not zero, the value in E is transferred to the Accumulator (Register A). Then C is added to HL - the instruction ADD HL,BC has to be used to do this because you cannot add a single register to a register pair. The Accumulator is then decreased by one and, if it equals zero, the computer jumps to the printing routine. Otherwise, the addition process is repeated until the Accumulator does equal zero.

```
        ORG &8000       ; the address where we want to locate this routine

        LD E,255        ; the first number to be multiplied
        LD C,123        ; the second number to be multiplied
        LD HL,0         ; to be used to hold the result
        LD A,E          ; A = E
        CP 0            ; compare A (and therefore E) with 0
        JP Z,print      ; if A does equal 0, jump to the printing routine
.loop   ADD HL,BC       ; Add BC to HL - because B equals 0 this is equivalent
                        ; to just saying, add C to HL)
        DEC A           ; decrease A by one - as we made A equal to E earlier,
                        ; this is equivalent to decreasing the first number by
                        ; one - the first number is the number of times we are
                        : going to add the second number to itself
        JP NZ,loop      ; if A doesn't equal zero now, we know that we haven't
                        ; finished the adding process and so we go round to do
                        ; it all again
        CALL print      ; otherwise, we are going to call the routine to print
                        ; result for us
        RET             ; when all of that's done we can return to BASIC

.print  LD BC,10000
        CALL prtdig
        LD BC,1000
        CALL prtdidg
        LD BC,100
        CALL prtdig
        LD BC,10
        CALL prtdig
        LD A,L
        ADD A,&30
        CALL &BB5A
        RET

.prtdig LD A,&FF
.loop2  INC A
        SBC HL,BC
        JR NC,loop2
        ADD HL,BC
        ADD A,&30
        CALL &BB5A
        RET
```

The printing routines (called .print and .prtdig) are identical to those described in Issue Four of Print-Out on pages 7 to 9 and therefore no further explanation is necessary. This is just one way of multiplying numbers together in machine code; the more complex and slightly faster methods involve the use of the SHIFT and ROTATE instructions which we have not yet looked at. If space permits, we may write a shift version of our multiplication routine in the next issue. As well as that, we'll be covering division (which incidentally can be written using either the shift method or in a similar manner to these multiplication routines. As a challenge you could try & write a program to do the opposite of our number printing routine (ie the user types in a number and the program then converts it into a number that can be used in the above multiplication routine).

To run this program on your CPC, you'll need a program called an assembler and we included our own version on the program tape and disc for Issue Seven along with an instruction file for successfully entering and running machine code programs.



Technical Tips

MORE READERS QUERIES ANSWERED.....

by BOB TAYLOR

In Issue 8, I wrote a Screen Dump routine for the Artist Designer program from a few issues earlier. That routine gave flattened circles & I said that to produce circular ones would need many sheets of paper. Since then, I have discovered that the Amstrad DMP 2000 has a graphics mode which does in fact give correct proportions (called 'bit image plotter graphic mode') and it may be present on other printers (it uses the codes: ESC, "*", 5, plus a two byte value of the number of bytes to be sent). Unfortunately, this mode can only print 576 dots to a line, so making it impossible to reproduce the full width of the screen (640 dots) across an A4 sheet of paper. However, it can copy the screen onto the sheet sideways and it is on this basis that I have written an RSX to dump an entire screen in any MODE.

To give some representation of the different colours which may be present on screen, dot patterns of differing densities are printed to simulate a black and white reproduction. The routine automatically assigns 'dot patterns' to colours according to their shading (and not according to their PEN Nos).

Sometimes a screen uses a dark colour for the background and many Screen Dump routines are notorious for using up the ink in printer ribbons, so I've included a facility to print the PAPER 0 colour as the lightest possible 'dot pattern' if it should be needed; the patterns for the other colours are then adjusted automatically to accommodate this.

The syntax for the RSX to give PAPER 0 as the lightest is: |SHADUMP,anything
The syntax for the RSX in its normal form is: |SHADUMP

The BASIC loader below should be SAVEd after typing-in and before RUNning. It produces relocatable code, and the program contains instructions for its use.

PROGRAM

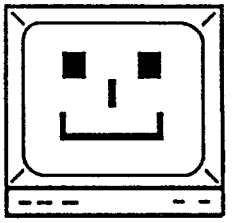
```
[F1] 10 'SHADED DUMP Loader  by Bob Taylor  (copyright 1991)
[20] 20 MEMORY &7FFF:RESTORE:PRINT:PRINT"Please wait a few seconds"
[D2] 30 FOR lin=0 TO &1C0/8-1:total=0:FOR n=0 TO 7:READ a$
[A2] 40 byte=VAL("&"+a$):POKE &8000+lin*8+n,byte
[4B] 50 total=total+byte:NEXT n
[0D] 60 READ a$:IF VAL("&"+a$)<>total THEN PRINT:PRINT"Error in line"lin*10+110
        :PRINT:END
```

```
[C4] 70 NEXT lin
[AB] 80 PRINT:PRINT"All M/C loaded":PRINT:PRINT"Press S to save M/C as
       SHADUMP.BIN":PRINT"or any other key to continue":WHILE INKEY$="":WEND:
       IF INKEY(60)<>-1 THEN SAVE "SHADUMP.BIN",B,&8000,&1C0
[4E] 90 PRINT:PRINT"To Load and Initialise !SHADUMP RSX with a program present,
       just use:":PRINT"MEMORY HIMEM-&1C0:a=HIMEM+1:LOAD"CHR$(34)"SHADUMP.BIN"
       CHR$(34)",a:CALL a":PRINT"with the Disc or Tape inserted"
[EA] 100 END
[CC] 110 DATA 21,21,00,19,06,13,7E,23,115
[67] 120 DATA E5,66,6F,19,7E,83,77,23,36E
[80] 130 DATA 7E,8A,77,E1,23,10,EF,01,383
[0C] 140 DATA 47,00,EB,36,C9,23,C3,D1,3E8
[C1] 150 DATA BC,18,00,47,00,4F,00,5C,1C6
[EE] 160 DATA 00,79,00,94,00,98,00,AE,253
[15] 170 DATA 00,BC,00,BF,00,C5,00,DA,31A
[97] 180 DATA 00,2A,01,30,01,40,01,51,0EE
[99] 190 DATA 01,59,01,5F,01,66,01,6A,18C
[B0] 200 DATA 01,F5,CD,1A,BC,78,32,7D,3C0
[A4] 210 DATA 01,FE,04,20,01,87,87,47,279
[D8] 220 DATA 4F,F1,C5,21,05,00,B7,3E,320
[9A] 230 DATA 1B,20,0A,79,90,C5,E5,CD,3C5
[A6] 240 DATA 35,BC,78,E1,C1,77,23,23,3C8
[E9] 250 DATA 23,23,10,EF,36,FF,AF,4F,378
[F0] 260 DATA 21,05,00,04,BE,23,20,02,12D
[A7] 270 DATA 71,04,23,23,23,CB,7E,28,24F
[0C] 280 DATA F3,0D,0C,10,FD,3C,FE,1C,36F
[EB] 290 DATA 38,E6,C1,11,06,00,C5,3A,2F5
[22] 300 DATA 7D,01,FE,02,1A,4F,28,05,214
[B0] 310 DATA 30,06,87,81,4F,87,87,81,31C
[4C] 320 DATA 87,87,4F,06,00,21,7F,01,204
[A0] 330 DATA 09,0E,04,ED,B0,C1,10,DE,367
[0E] 340 DATA 11,7F,02,01,72,01,CD,54,227
[C6] 350 DATA 01,21,8F,01,3A,7D,01,D5,23F
[BE] 360 DATA 01,00,00,CB,4D,D5,E5,C5,398
[00] 370 DATA F5,CD,F0,BB,87,87,4F,06,4D0
[D0] 380 DATA 00,21,06,00,09,F1,20,02,143
[2C] 390 DATA 23,23,C1,F5,FE,02,28,33,357
[15] 400 DATA 30,3B,C6,07,CB,60,28,11,29C
[B8] 410 DATA 1F,CB,68,28,0A,1F,CB,70,2DE
[52] 420 DATA 28,03,1F,CB,F8,CB,F0,CB,493
[66] 430 DATA E8,CB,E0,F5,A6,B1,4F,F1,61F
[05] 440 DATA 23,A6,B0,47,CB,78,20,18,33B
[72] 450 DATA F1,E1,D1,1B,FE,02,38,B3,4A9
[CD] 460 DATA 1B,18,B0,3E,0C,CB,70,28,290
[C4] 470 DATA DC,3E,03,18,D6,4E,23,46,2C2
[83] 480 DATA 79,CD,5E,01,78,E6,0F,CD,3DF
[AE] 490 DATA 5E,01,F1,E1,D1,D1,2B,2B,429
[15] 500 DATA CB,7C,28,8B,06,70,AF,CD,3EC
[BA] 510 DATA 5E,01,10,FA,1B,1B,1B,1B,1D5
[07] 520 DATA CD,09,BB,FE,FC,C8,CB,7A,598
[7C] 530 DATA CA,BB,00,C9,0A,CB,7F,C0,462
[B3] 540 DATA CD,5E,01,03,18,F6,32,7E,2ED
[77] 550 DATA 01,CD,2B,BD,D8,3A,7E,01,347
[48] 560 DATA 18,F7,53,48,41,44,55,4D,2D1
[FB] 570 DATA D0,00,0A,0D,1B,41,04,1B,162
[81] 580 DATA 2A,05,00,02,FF,00,00,0F,13F
[BF] 590 DATA 0F,0F,0F,0F,0A,0F,0E,0A,06D
[0A] 600 DATA 0F,0A,0F,0B,0E,07,0D,07,05C
[44] 610 DATA 0A,07,0F,0B,0E,0B,0E,0A,05C
[E4] 620 DATA 0E,07,05,01,0E,01,0E,0C,044
[F1] 630 DATA 03,0C,03,03,0A,05,0C,0A,03A
[75] 640 DATA 05,0A,05,04,01,08,02,08,02B
[5A] 650 DATA 02,08,02,0A,00,0A,00,08,028
[E4] 660 DATA 00,02,00,00,00,00,00,00,002
```

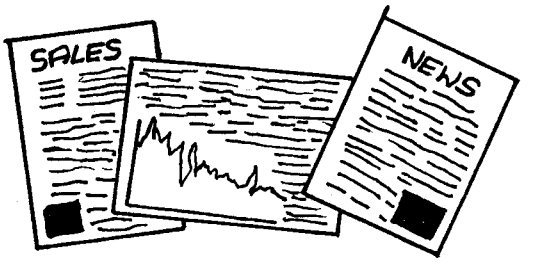
Linechecker

A PROGRAM TYPING AID

All programs in Print-Out have Linecheck codes which are enclosed in brackets at the start of a line. Don't enter them in as they're designed to be used with Linechecker to eliminate errors when typing in programs which appear in this magazine. Please note, all programs will run whether Linechecker is being used or not. For information on how to use Linechecker, please see Issue Three.



CPC News



Helpline

Angela Allum has been running a help service for owners of CPC computers for about the past twelve months and, intrigued by her adverts in Amstrad Action's Small Ads, I wrote to find out what it was she did. So here is Angela's reply:

'My service is part serious, part fun. I type in listings from the Amstrad CPC magazines, some from 1985 & do a Special Offer of any three games or the Fractal Set from ACU on disc or tape for £10, any extra listings cost another £1.

'I can provide details of the screen addresses & colour codes. On the fun side I provide some pokes for games, and help with games I have solved or enjoyed and made a lot of progress in.'

If this appeals to you, then you can get in touch with Angela:

Angela Allum, PO Box 116, Bracknell, RG12 4PQ

SUPERWIMP

Some of you may have seen the review of Tearaway from CPC Network in Amstrad Action recently (nice to know we had already reviewed it six months earlier) and I have recently heard from the owner of CPC Network, James Verity.

Since we reviewed Tearaway in Issue Six, extra help files have been added to enable even the novice to find pokes easily and effectively. I am also told that the actual program has been improved and made even better. With these changes, Tearaway represents really excellent value-for-money.

In the past few months, CPC Network have also sent me a copy of Superwimp, an easy-to-use WIMP environment. The program provides you with about 20 commands in the form of RSXs which can be used from within BASIC to produce and run your own custom-designed menu system.

The system is very friendly and anyone with a working knowledge of BASIC, and a bit of patience, should be able to produce professional looking frontends for their programs.The best things about this piece of software are its ease of use, and the fact that all of the tricky bits of programming have been done for you. All you have to do is to supply the creativity and imagination.

The pointer and menus can be controlled by keyboard, joystick or mouse. Also provided are various 'designers', which allow you to change the characters and icons (some are supplied ready for use with the program). To help you get into the program, a demo and plenty of helpfiles are supplied, and the whole program is very nicely put together.

However, I wonder whether many people will get a great deal of use from this program. After all, you'll probably only use a WIMP environment on software for use by other people - any programs you produce using Superwimp are the copyright of CPC Network and cannot be sold or published - and I wonder whether it's worth shelling out for this type of program. Still, if you want a system which allows you to add a professional frontend onto your programs, you will not find a more polished program anywhere. For more details, see their advert elsewhere in this issue or contact them at: 3 The Cottons, Outwell, Wisbech, Cambs PE14 8TL.

SOFTWARE REVIEWS

WORD PROCESSORS

Many readers have suggested that Print-Out should run a section on commercial programs. As from this issue, we'll be reviewing as much serious software as we can. along with tips and hints for these utilities. This issue, we'll be looking at the most popular word processing programs for the CPCs: Brunword, Protext and Tasword. Every piece of serious software has its fans and critics and while some people may hate one program, others may love it. As the magazine is produced on a CPC with the Protext suite of programs. we've asked some readers to give their views on the programs under test. in order to provide an unbiased review.

Protext from Arnor (Price: £19.95 on tape, £26.95 on disc, £39.95 on ROM)

Protext features all of the usual text editing commands that you would expect from a word processor, such as Centre, Right-Justify, Block Move/Delete/Copy and a selection of functions to set the margins and tabs (including a decimal tab to align numbers correctly). It also allows you to have an exceptionally long line length and to return to a marked position in the text.

In addition to these the program has an excellent system of moving the cursor around the document (and also very quickly!). Against this, it doesn't allow you to retrieve deletions, and there are times when the cursor does not do what you might have expected it to. Also page breaks are not indicated on the screen, and it does not automatically re-justify when you insert extra text into the middle of a line.

Perhaps Protext's strongest point is its means of controlling the printer. It allows you to insert pre-defined 'printer control codes' which perform specific tasks, such as italics or bold typing, and if you need to send more detailed or complicated information to the printer, it has its set of 'stored commands' that are inserted directly into the document.

If you are thinking of ever writing in a foreign language (including German, French and Danish) then Protext supports all of the funny little accents without any trouble at all, and if your printer can reproduce these characters, Protext sets it up for which ever language you have chosen.

When in the Command Mode there is a host of instructions available, including plenty for file handling & printing, and options to change much of the program. Protext also has a good Search and Replace routine which is extensive and allows you to insert wildcards in your strings and even printer codes! The Command Mode allows you to use RSXs or to enter BASIC (eg. for changing key definitions), but unfortunately blocks part of the text area when using it.

Without a doubt Protext's greatest drawback is its spell checker - this needs an extra ROM (or disc) to be purchased and the dictionary is held on a disc, so making any spellchecking unbearably slow.

Brunword 6128 from BRUNNING SOFTWARE (Price: £30.00 on disc)

Brunword, like most of the other popular word processors on the CPC, has been around for several years now and is available in several incarnations. Although we are reviewing only the standard program in this article, it may be worth also looking at the added features of some of the top-of-the-range versions.

Brunword's greatest advantage over its competitors is its 'built-in' spelling checker. Because the spellchecker & dictionary are loaded with the main program, it means that Brunword is exceptionally quick on any spellchecking - however, it does mean that the program will only run CPCs with at least 128K of memory.

Brunword contains most of the editing commands of Protext and in addition, features 'true' justification - when you add a word in the middle of an existing line, the paragraph automatically reformats to accommodate the new text. Also it allows you to retrieve anything that you have deleted, and it will hold files or part-files in memory. When typing, the cursor operates only on the text present and this can be very beneficial.

However when deleting characters, the cursor often disappears and this can be annoying as it is harder to see what you are about to erase. Most printer codes take up a space (when printed) and so, for example, it's impossible to underline just a single letter in a word without leaving a space.

Page breaks and sub/superscript characters are shown on-screen which is helpful when trying to visualise a page layout. Against this, a miniature character set is used when the page is over 80 characters wide and this makes it very hard to read - added to this there's a poor colour combination for the program.

Files can be a maximum of 23K in length & also can be protected by a security code if desired. Brunword comes with a simple database (a bit like a card index) and this is greatly enhanced in Brunning's Infoscript program.

Tasword 6128 from Tasman Software (Price: around £25.00 on disc)

Tasword's greatest asset compared to the other two programs is that on a 6128 it utilises the extra banked memory to allow you to have a file of 60K. When you compare this to 39K of Protext and 23K of Brunword, you can see that this could be a great advantage to the serious writer who is composing a chapter of a novel and needs to be able to access anything that has been written already.

Another great boon are the four notepads which are available for you to store text, ideas, or anything else that you wish to remember. These are separate from the main article and so don't interfere with your piece of work.

Also, Tasword features a set of Mail Merge commands which allow you to create personalised letters or address labels - these can even be printed conditionally ie. letters will only be printed if they fulfil certain conditions.

Whilst Tasword is not as user-friendly as Protext, you can readily customise it to suit your own preferences, set up user-defined keys, and alter the program so that it fulfils your requirements.

Despite the fact that Tasword has the fewest features when compared with the other programs, Tasword can chain print files & display a prompt when printing, so allowing keyboard input to be printed if required. You can also retrieve the last line that was deleted and sub/superscript characters are displayed. To use foreign letters and special symbols, Tasword has a '2nd Character Set' which you can call up and access.

Unfortunately, Tasword only lets you move, delete or copy entire lines - when you move text you can only move it to the line above the cursor. The cursor also behaves strangely and runs through 'padding' spaces in short lines when erasing many characters. Although page ends are indicated, the program gets confused if any page throws have been used and no word count or page number is shown on the screen. Even though there are 'help' screens available from within the program, some of the keypresses were a bit awkward and difficult to remember.

Like Protext, the spellchecker (Taspell) requires a separate disc which costs around the £17.00 mark. When reviewing programs such as Tasword, it is important to bear in mind the utility discs which can enhance the program. Also worthy of mentioning, is the Tasword manual - it is clear, well explained and easy to use as a reference book when the basics have been mastered.

The Verdict....

All three word processors are available in different configurations and so it is worth bearing this in mind when trying to decide which one to buy. Also it is important to remember what *you* want from a word processor, otherwise you may end up choosing one which either does not fulfil your needs, or is too complex.

The most important thing when buying any piece of software is whether it will let you do what you want & with the minimum of fuss. With a word processor this means, when the program is loaded can I get straight on with the job in hand and are all of the common instructions quickly, and easily, accessible.

In conclusion, Protext is exceptionally fast and extensive. As it comes with a well written manual, most people should find it relatively easy-to-use, and it certainly will be able to handle anything you choose to do with it. On the other hand, I feel that Brunword is not so intuitive to use as Protext, although many will probably disagree with me. Most reviewers are ready to slam Tasword firmly into third place and class it as a poor man's word processor - this isn't really fair on either Tasword or its users; although it's slower than the others & does have less features, it does have several points in its favour, such as the large text files and notepads.

If you don't already own a word processor, I would recommend Protext with its many features. If you already use one, then unless you're determined that learning to use a different word processor is better than ploughing on with the fewer features your current one offers, I suggest that sticking with your present one will be better than the idiosyncrasies of a new one.

Next issue, we will be looking at some of the programming utilities which are currently available for the CPC user.

030 &BB5A TXT OUTPUT
ACTION: Output a character or control code (&00 to &1F) to the screen
ENTRY: A contains the character to output
EXIT: All registers preserved
NOTES: Any control codes are obeyed; nothing is printed if VDU is disabled
Characters are printed at the current cursor position

031 &BB5D TXT WR CHAR
ACTION: Print a character at the current cursor position
ENTRY: A contains the character to print
EXIT: AF,BC,DE,HL are corrupt; all others are preserved
NOTES: Any control characters are printed and not obeyed

032 &BB60 TXT RD CHAR
ACTION: Reads a character from the screen at the current cursor position
ENTRY: No entry conditions
EXIT: A contains the character read, carry true; or carry set false
NOTES: This call actually uses the TXT UNWRITE indirection

033 &BB63 TXT SET GRAPHIC
ACTION: Enables or disables graphic character writing
ENTRY: To switch on - A must be non-zero; to switch off - A must be zero
EXIT: AF corrupt; all other registers preserved

034 &BB66 TXT WIN ENABLE
ACTION: Sets the boundaries of the current text window
ENTRY: H holds the column of an edge, D holds the column of the other edge
L contains the row of an edge, E contains the row of the other edge
EXIT: AF,BC,DE,HL corrupt

035 &BB69 TXT GET WINDOW
ACTION: Returns the size of the current window
ENTRY: No entry conditions
EXIT: H contains the number of the left column, D holds the right column;
L contains the number of the top row, E holds the bottom row; if it
covers the whole screen, Carry is false; A is always modified

036 &BB6C TXT CLEAR WINDOW
ACTION: Clears the window; moves cursor to top left corner of window
ENTRY: No entry conditions
EXIT: AF,BC,DE,HL are corrupt; all others are preserved

037 &BB6F TXT SET COLUMN
ACTION: Sets the cursor's horizontal position
ENTRY: A contains the column number to move the cursor to
EXIT: AF and HL are corrupt; all others are preserved

038 &BB72 TXT SET ROW
ACTION: Sets the cursor's vertical position
ENTRY: A contains the row number to move the cursor to
EXIT: AF and HL are corrupt; all others are preserved

039 &BB75 TXT SET CURSOR
ACTION: Sets the cursor's vertical and horizontal position
ENTRY: H contains the column number; L contains the row number
EXIT: AF and HL are corript; all others are preserved

040 &BB78 TXT GET CURSOR
ACTION: Gets the cursor's vertical and horizontal position
ENTRY: No entry conditions
EXIT: H holds the column; L holds the row; A has the roll count
NOTES: The roll count is increased if the screen is scrolled down, and is decreased when the screen is scrolled up

041 &BB7B TXT CUR ENABLE
ACTION: Allows the text cursor to be displayed (controlled by user)
ENTRY: No entry conditions
EXIT: AF is corrupt; all others are preserved

042 &BB7E TXT CUR DISABLE
ACTION: Prevents the text cursor from being displayed (controlled by user)
ENTRY: No entry conditions
EXIT: AF is corrupt; all others are preserved

043 &BB81 TXT CUR ON
ACTION: Allows the text cursor to be displayed (controlled by system)
ENTRY: No entry conditions
EXIT: All registers and flags are preserved

044 &BB84 TXT CUR OFF
ACTION: Stops the text cursor from being displayed (controlled by system)
ENTRY: No entry conditions
EXIT: All registers and flags are preserved

045 &BB87 TXT VALIDATE
ACTION: Checks whether the cursor's position is within a window
ENTRY: H contains the column to check; L contains the row to check
EXIT: H holds the column where the next character will be printed, and L holds the row; AF corrupt; if printing will not cause the window to scroll then Carry true and B is corrupt; if printing will cause the window to roll up then Carry is false and B holds &FF; if printing will cause the window to roll down then Carry is false & B has &00

046 &BB8A TXT PLACE CURSOR
ACTION: Puts a cursor blob on the screen at the current cursor position
ENTRY: No entry conditions
EXIT: AF is corrupt; all others are preserved
NOTES: TXT OUTPUT and TXT SET CURSOR automatically reposition the cursor when they are called

047 &BB8D TXT REMOVE CURSOR
ACTION: Takes a cursor blob off the screen at the current cursor position
ENTRY: No entry conditions
EXIT: AF is corrupt; all others are preserved
NOTES: See TXT PLACE CURSOR for more information

048 &BB90 TXT SET PEN
ACTION: Sets the foreground PEN colour
ENTRY: A contains the ink number to use
EXIT: AF and HL are corrupt; all others are preserved

BASIC Tokens (3)

In my previous article on BASIC Tokens, we looked at how the Operating System used the area of memory designated for storing a Program. This month I intend to cover its use of two further areas: those used for storing Variables and Arrays. Just to recap from the last article, all two byte values, distances or addresses have the low byte before the high byte.

VARIABLES AREA

This ought to be called the 'Variables and DEF FN Function Names' area but as that's a bit of a mouthful, we will shorten it. Variables and Function names are treated fairly similarly so we'll look at both at the same time. For convenience we'll use the term 'Name' for both a Variable and a Function Name unless we need to be specific about one or the other.

When the BASIC Parser encounters a Name in a Program line, the Variables area is searched, probably from the start, to find whether an identical entry already exists (the case of characters is ignored). If so, the old entry is amended with new data (see section on Data below).

When a new Name is being used, it is stored in the area above the last entry, if there is one. Function Names are quite freely mixed in with Variables and no attempt is made to keep the two sorts separate; also the Names are not arranged in alphabetical or any other order. The Arrays area is moved upwards without the loss of contents and the System Variables: Array start and Free Area start addresses are amended to their new values.

Each Name entry consists of a sequence of bytes as follows:

TWO DISTANCE BYTES: The use of these bytes is quite strange. Type is disregarded except for a difference between Variables and Function names.

As far as Variables are concerned, every first occurrence of a Name starting with a so far unused character has these bytes zeroed. Each subsequent entry with the same initial letter has the distance from the byte before the first character of the Name of the **previous** entry with this initial letter to the first byte in this area.

The first **Function** name has the bytes zeroed, then any further ones (regardless of initial letter) have the distance from the byte before the previous Function name's initial character (again regardless of letter) to the first byte in the area as for Variables. Why every entry does not have a distance is beyond me. It also puzzles me as to why subsequent entries point back to previous occurrences and not the other way around - it is almost as if the Variables area is searched from the end back to the beginning.

NAME BYTES: Next are the characters of the name with bit 5 reset (this makes the letters upper case only and so speeds up the search. However numbers and the full stop are also converted, resulting in quite strange values). To signal the end of the name, the last character has bit 7 set. Names can have up to 40 characters.

VARIABLE TYPE BYTE: The next byte indicates which type of Variable we have:

&01 for **DEFINT (%) Variable**
&02 for **DEFSTR ($) Variable**
&04 for **DEFREAL (!) or undefined Variable (= Real)**
&41 for **DEFINT (%) Function Name**
&42 for **DEFSTR ($) Function Name**
&44 for **DEFREAL (!) or undefined Function Name**

Function Names have the same Type Byte as the corresponding Variable version, except that they have bit 6 set. However, compare the value used for '%' and '$' Variable types with the Tokens given in **TABLE 1** of the 'Tokens Article' and note the differences. I can only surmise that it was originally intended that Program Tokens should be the same as the Type Bytes in the Variables area (you'll notice that each Type Byte has only one bit set for easier checking) but that for speed of checking for the end of a statement marker, the colon character was converted to a byte of &01, so displacing the Program Tokens for % and $ Variables upwards to their present positions.

However, if Type Bytes and Tokens of &02, &03 and &05 had been used, as well as being as easy to check as those which are used, they could also have doubled as lengths for the remainder of each entry.

DATA BYTES: The last bytes are the relevant data for each of the Name types:

INTEGER VARIABLE: Two bytes for the value

STRING VARIABLE: One byte for the length of the string (strings can only be up to 255 characters long) and then:
Two address bytes giving the location of the start of the string whether it is in the Program or Strings Area. Note that the actual string is not stored in the Variables area. The length byte followed by the address of the string is called the 'String Descriptor'. This may be familiar to you in connection with passing parameters to and from RSXs. The address obtained from the RSX string parameter will be that of the length byte in a String Descriptor here in the Variables area, or as an element in the Array area; additionally on a 6128 only, it may be in the Operating System's upper workspace. The address part of the String Descriptor will point to the actual string in the Strings Area.

REAL VARIABLE: 4 bytes for the Mantissa (low byte first) with bit 7 of byte 4 set if the number is negative. Unlike numbers held in the Program area, here both positive and negative numbers have to be expressed in 5 bytes only - no negative Token & no number Token either.
This is then followed by: One byte for the Exponent

FUNCTION NAMES (of all types): Two byte address of the first valid character (that is, discounting spaces) after the DEF FN Function Name as it occurs in the Program. This character should be, either the opening bracket before any Local variables, or the equals sign if there are none - Local Variables are those Variables listed within brackets following the DEF FN Function Name in the Program. They are 'local' because any values or strings which are passed temporarily to them for the DEF FN to work upon will not effect any existing values or strings already held by those variables before the FN was called; any such values or strings will be restored on completion of the FN).
If part of a Program is deleted either before the DEF FN or even the DEF FN itself, this address isn't amended (until RUN) and so may no longer point to the token.

Type in LISTING 2 below and use it to confirm the above details for yourself; you can add further lines, numbered between 50 and 90, containing other examples of Variables. The screen display will show the following:

1. Each Variable is printed out on a new line. It begins with the address of the low 'distance' byte (for reference only).
2. Then comes the name of the variable but without anything (& % !) to define it, and this is followed by all the bytes of the entry in hexadecimal form but without the preceding '&' for clarity. These hexadecimal bytes go from the distance bytes through to the last value or string descriptor byte.

NB: For the String Variable entries, the address part of the String Descriptor will not point to the Strings area if the String Variable (or Array element - see later) is being assigned a single delimited string to be found in the Program or Direct Input area which is where the address will indicate - see next issue for those Strings which will be stored in the Strings area).

LISTING 2:

```
[6C] 1 MODE 2:INK 0,13:INK 1,0:DEFINT a:a=1:a$="AB":DIM a(3,3):a!=@a$
[09] 10 b$="123":joined$="J"+a$:sliced$=LEFT$(joined$,2)
[43] 20 b=65535:b%=&FF:aaa=-32767
[84] 30 DEF FNa=b%*10:DEF FNb(a)=a^a:DEF FNa$(b)=STRING$(b,"a")
[89] 40 PRINT FNa,FNb(3),FNa$(5)
[9D] 1100 DEFINT n
[BB] 1110 FOR n=@a-4 TO @n-5:a=n
[4D] 1120 PRINT:PRINT "&"HEX$(n,4)" ";
[45] 1130 PRINT CHR$((PEEK(n+2)AND &7F)-32*((PEEK(n+2)AND &7F)<65));
[C9] 1140 n=n+1:IF PEEK(n+1)<128 GOTO 1130
[62] 1150 PRINT,
[3A] 1160 FOR n=a TO n-(-1+PEEK(n+2)AND 7)*(PEEK(n+2)<&40)+4
[DF] 1170 PRINT " "HEX$(PEEK(n),2);
[8B] 1180 NEXT:n=n-1
[5A] 1190 NEXT:PRINT
```

ARRAYS AREA

When it finds a DIM statement, the Parser seeks to install a new Array here, after any that already exist. However, an array which is already in use can't be reDIMensioned without producing the error 'Array already dimensioned'.

Unlike variables which will always exist once set up (unless CLEAR is used), arrays can be removed by using ERASE and the Array area will be reduced by the space previously taken up by the erased Array. There are no Array equivalents to DEF FN Function Names but all three Variable types of Array are possible.

The sequence of bytes for each Array entry is:

TWO DISTANCE BYTES: Strange again! The first entry of each type has zeroed bytes (initial letters are not significant in this area). Subsequent occurrencies of the same **type** have the distance from the byte before the first character of the name of the previous entry of the same type to the first byte in the Array area. (cf Variables; I really think someone's pulling my leg!)

NAME BYTES: Next comes the name of the Array, where the rules already given for Variables also apply.

TYPE BYTES: Similarly, the type byte which follows is the same as for variables; (No DEF FN's).

LENGTH OF ARRAY: These 2 bytes hold the remaining length of this Array starting from the Number of Dimensions byte which follows.

One byte for the **NUMBER OF DIMENSIONS**, so there can be up to 255 dimensions. This is then followed by pairs of bytes which give the **SIZES OF THE DIMENSIONS**, starting with the last Dimension if there are more than one, and continuing through any others to the first. The number of pairs of bytes that are used equal the number of dimensions. Dimensions can be of any size provided that there is enough room available in memory. The size of a dimension is always one more than that written in the DIM statement because the lowest numbered element there starts at 0 and not 1.

DATA ELEMENTS: Lastly, we've the storage of data elements. The data is stored in these in 2, 3 or 5 bytes in exactly the same way as with the Variables – ie numbers only are stored, with strings having the string descriptor pointing to the required string. However, we now have whole collections of values or strings requiring storage so a fixed sequence is used as follows:

```
With a 1 dimension array having (x):      (0) (1) (2) etc.  up to (x)
With a 2 dimension array having (x,y):    (0,0) (1,0) (2,0) up to (x,0)
                                    then  (0,1) (1,1) (2,1) up to (x,1)
                          and so on up to (0,y) (1,y) (2,y) up to (x,y)
With a 3 dimension array having (x,y,z): (0,0,0) (1,0,0) (2,0,0) up to (x,0,0)
                                    then         (0,1,0) (1,1,0) up to (x,1,0)
                          and so on up to        (0,y,0) (1,y,0) up to (x,y,0)
        then (0,0,1) and via the same sort of sequence as before up to (x,y,1)
                          and then repeating over and over again up to (x,y,z)
```

On first DIMensioning an array, all the element bytes are filled with zeros, either to give null strings (ie of zero length), or to give 0 as the contents of number Array elements both Integer and Real. Only as any element is used are its data contents changed. The length of an Array doesn't vary from when it is first DIMensioned to when all of its elements are in use. However with a String Array, extra space has got to be allowed for some of the strings associated with it to be stored in the Strings area.

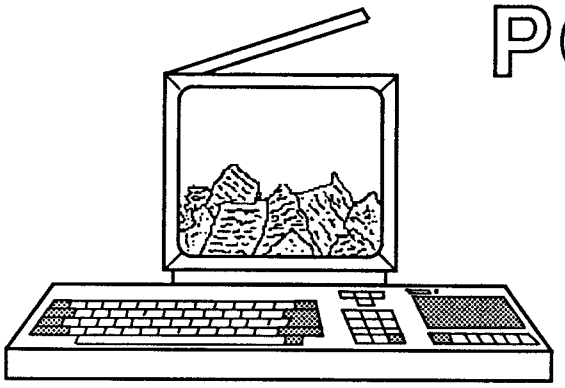
Using LISTING 3 below, you can add your own lines 80 & 90 containing examples of arrays, and you can obtain a display similar to that for Variables above; the sequence will be:

1. Address of the low 'distance' byte at the start of each line
2. Followed by the name of the Array (without any type details)
3. Followed by all the bytes relevant to the entry.

LISTING 3:

```
[6C] 1 MODE 2:INK 0,13:INK 1,0:DEFINT a:a=1:a$="AB":DIM a(3,3):a!=@a$
[1B] 50 FOR a=0 TO 3:FOR b%=0 TO 3:a(a,b%)=a+b%*&100:NEXT:NEXT:b%=&FF
[56] 60 DIM b(1):b(1)=1
[C6] 70 DIM q$(2):q$(0)="1st":q$(1)="2nd":q$(2)="3rd"
[94] 1200 DEFINT n:DIM n(0)
[AA] 1210 FOR n=@a(0,0)-11 TO @n(0):a=n
[7E] 1220 PRINT:PRINT"&"HEX$(n,4)" ";
[47] 1230 PRINT CHR$((PEEK(n+2)AND &7F)-32*((PEEK(n+2)AND &7F)<65));
[EE] 1240 n=n+1:IF PEEK(n+1)<128 GOTO 1230
[64] 1250 PRINT,
[56] 1260 FOR n=a TO n+4+PEEK(n+3)+256*PEEK(n+4)
[E1] 1270 PRINT " "HEX$(PEEK(n),2);
[8D] 1280 NEXT:n=n-1
[5C] 1290 NEXT:PRINT
```

In the next issue of Print-Out, we will be completing our look at BASIC's use of Memory by considering the area designated for holding strings.



POKING AROUND

Lift the lid on your CPC

Many thanks to those readers who have helped in producing this column. Remember if you have any hints or tips then please send them in to us at the usual address.

Multiface Tips

Here's a neat little trick for Multiface owners, from Alastair Henderson. It completely hides your Multiface from any game that you are trying to load - very useful for those programs which can still detect the Multiface even when it has been turned off (as mentioned in the Multiface manual). Alastair explains:

'Press down the red button (STOP) and keep it held down, then press down the green button (RESET) and let go of it, then release the red button. Hey presto, the Multiface is completely undetectable by any protection scheme, as even the Multiface itself doesn't realise that it's there - try loading a multiface-saved game!'

This seems to work to various degrees on different machines. On my Multiface, the border flashes when you press 'R' to leave the menu; on other computers, the machine just resets - however, it does seem to have made it totally invisible!

Another hint for the Multiface is: When you have the menu bar on the screen, press the 'f0' key (or zero on the numeric keypad) and this will tell you which version of the Multiface you have got.

BASIC Hints

Enough of all that Multiface magic, it's time for some BASIC tips, sent in by Richard Wildey from East Sheen, London.

Firstly on a 6128 only, use the following line which displays the incomplete listing of a partly loaded BASIC program - either because there has been a 'READ ERROR' or a 'READ FAIL':

```
POKE &170,PEEK(PEEK(&B11C)*256+PEEK(&B11B))
```

While not a new tip, some people may not know that there is an RSX present in your CPC from the moment you switch it on. This is |BASIC and completely resets the computer when used - I only include this because it is not mentioned in the 6128 manual. Also when running a file do this to save time: Type the filename in and then hold down CONTROL and press LEFT CURSOR, followed by the small ENTER.

Finally, you can use fonts designed with Art Studio (& possibly with Advanced Art Studio). Type:

```
SYMBOL AFTER 32:h=HIMEM+1:LOAD "FILE.FNT",h
```

To get the font back to normal, just get a 'DISC MISSING' error message.

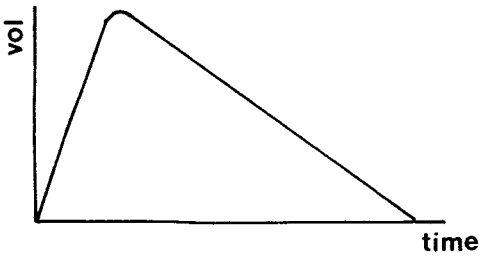
Many thanks Richard for all of those tips. Hopefully, next issue we will have some more things for you to try out on your CPC, and will be back to our normal size of two pages.

SOUND

An introduction to the CPC's sound chip
PART 2

Last issue we looked at the Volume Envelope (ENV) and investigated the numbers which can follow it. This issue, we are going to look at some complicated volume envelopes, design envelopes for a few instruments and even start to compose some simple tunes.

The volume envelope that we were using last time had four numbers (parameters) following it. This allowed us to change the volume by making it either louder or quieter, at a uniform rate. However, in real life no sound changes its volume in a perfectly regular way - for example a piano rises to its maximum volume fairly quickly and then returns to silence a little bit more slowly. If this were drawn as a graph of time against volume, it would look something like this:



Using the envelopes we looked at last time, we wouldn't be able to mimic this sound as it has 2 very distinct parts - the volume rising and then decreasing again. However, last time we were looking at greatly simplified volume envelopes. These are the parameters following the complete volume envelope:

ENV N,P1,Q1,R1,P2,Q2,R2,P3,Q3,R3,P4,Q4,R4,P5,Q5,R5

Although this may seem complicated, it is actually quite easy to understand if you break it down into five sections. The letter 'N' would be a number from 1 to 15, and tells the computer which envelope we wish to define.

Each set of P,Q,R can be treated as a different part of the envelope (ie. each set of three letters tells the computer to do a different thing to the volume of the sound). Each part, or section, controls the volume of the sound for a while, and then hands control over to the next section. Thus, during the course of one single note, you can make upto five separate changes to its volume.

You don't actually have to use all five sections, but each section that you do use must be complete. So if you use a section, all three parameters (P,Q,R) must be present. The letters mena the same as in the envelopes in the last issue:

P is the STEP COUNT - this is the number of steps that there are in this section of the envelope and can be a number from between 0 and 127.

Q stands for STEP SIZE - this is the amount that the computer is to increase or decrease the volume by in each step in this particular section. It can have a value of between -128 and 127.

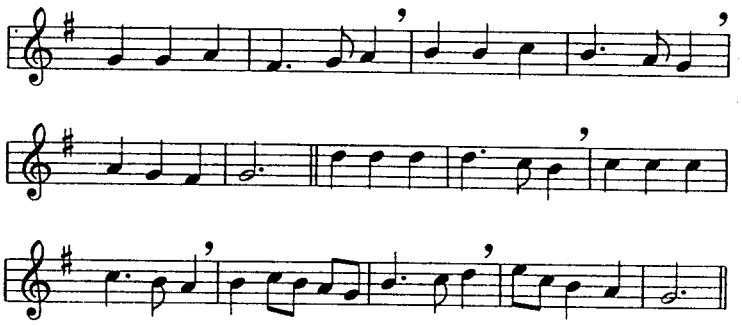
R is the PAUSE TIME and tells the computer how long each step in this section is to last. It is measured in hundredths of a second, and can be a value between 1 and 255.

These multiple section volume envelopes don't really need a great deal explained as they are just like having upto five single section volume envelopes rolled in one. The same rules as before apply when designing them, only your sounds can be more complicated. As an example of what can be done, here are some envelopes and alongside each are the musical instruments which they might pass for:

ENV 1,1,15,1,3,-1,4,12,-1,8	HARPSICHORD
ENV 1,2,7,1,1,1,3,5,-3,6	PIANO
ENV 1,1,15,1,5,-3,2	WOODBLOCK
ENV 1,3,5,1,5,-1,3,1,0,9,1,-10,1	TROMBONE
ENV 1,3,5,1,2,-1,1,3,-1,6,2,-5,1	OBOE
ENV 1,1,15,1,0,3,1,3,-5,1	SNARE DRUM

COMPOSING MUSIC

It's now possible to compose a fairly simple tune and, what's more, be able to recognise it!!! The piece of music that I have chosen to massacre in the painful rendition below is the National Anthem (mainly because it was the only piece of music that I could find, and I would be able to know if it sounded vaguely like it was supposed to!)



As I know nothing about music, I am not going to even attempt to explain about any of the notes in the piece, the time signature, the key or anything else; but we have still got to get the notes into the computer and play them.

First we need to work out which notes are being played. If my memory serves me correctly, the notes in the piece are as follows:

GGAFGABBCBAFAGFGDDDDCBCCCCBABCBAGBCDECBAG

In the back of the CPC manuals, there's a chart which allows you to convert a note into a 'period' or pitch for the SOUND command. Thus middle 'C' has a pitch of 478 - I feel it sounds better if it's played one octave above Middle C and so this gives rise to the figures in the DATA lines of the following program.

You must also specify the duration of the notes and, as far as I can remember, these are the values of the various notes, relative to one another:

SEMIBRIEVE	32	
MINIM	16	and 'dotted' notes are half as much again as their
CROTCHET	8	standard counterpart - and so a dotted minim has a
QUAVER	4	duration of 24.
SEMI-QUAVER	2	

All of the information about the pitch of the note & its duration are stored in the DATA statements at the end of the program (where it's easy to alter if there is a mistake in it). In all of my musical programs, I store the various bits of information in the following way: note pitch,note duration

This data is then read by lines 10, 20 and 40 and the sound is then played by line 30 at maximum volume (volume 15). You may be wondering why the duration has to be multiplied by 10 - the answer is that the note durations in the data are relative values & needs to be multiplied by a number that is related to the time signature of the piece of music. Unfortunately, I have never understood anything about time signatures and so just fiddled about with the number until it sounded OK to me. If you feel it should be faster or slower, then just alter the number following 'dur' until it suits you.

```
10 FOR i=1 TO 41
20 READ pitch,dur
30 SOUND 1,pitch,dur*10,15
40 NEXT i
100 DATA 319,8,319,8,284,8,358,12,319,4,284,8
110 DATA 253,8,253,8,239,8,253,12,284,4,358,8
120 DATA 284,8,319,8,358,8,319,24,213,8,213,8
130 DATA 213,8,213,12,239,4,253,8,239,8,239,8
140 DATA 239,8,239,12,253,4,284,8,253,8,239,4
150 DATA 253,4,284,4,319,4,253,12,239,4,213,8
160 DATA 190,4,239,4,253,8,284,8,319,24
```

As you will have noticed, some of the notes have the same pitch as the following note, and so the two run together producing one long note. There are two ways of preventing this:

1) Playing a very short note (with a duration of '1' and a pitch of '0', so it cannot be heard) in between the two notes with the same value.
2) Using an envelope which increases the volume from zero, and then decreases it back down to zero at the end of a note. In this piece of music I'm going to use the second method, which can be implemented by simply adding these 2 lines. It also makes the CPC sound more realistic:

```
5 ENV 1,3,5,1,1,0,8,1,-2,4,13,-1,8
30 SOUND 1,pitch,dur*10,0,1
```

To make the tune sound 'fuller' (that is, as if more than one instrument were playing the tune) we can play the same tune on a different channel, and the only difference will be that the piece of music will be played one octave lower. This can easily be achieved by adding:

```
35 SOUND 2,pitch/2,dur*10,0,1
```

We have now managed to create a quite respectable piece of music, although there is a long way to go before it is finished. In our next issue, we will be looking at the tone envelope & perhaps a few of the special commands associated with the sound chip.

READERS' SMALL ADS

PRESTON ROS BBS - Is now on-line 24 hours a day, 7 days a week - on 0772 652212. Speeds available are 300 - 1200/75 - 1200 - 2400. Protocol is 8 N 1 (scrolling). Preston ROS is a serious based BB for users of Amstrad CPC/PCW ranges of computers. Please give it a call soon!

WANTED - Any STOP PRESS PD fonts and clip art, etc. Will return any disc full of PD stuff - R Wildey, 41 Enmore Gardens, East Sheen, London SW14 8RF.

FOR SALE - Colour CPC 6128, Rombox and ROMs, Multiface II, 3.5 inch second disc drive, speech synthesiser and loads of other CPC hardware & software for sale. James Neill at 4 Oak Treee Meadow, Watton, Wakefield, West Yorkshire WF2 6TF, or phone (0924) 251608 after 7pm.

PLAYMATES - Issue 8 out now at £1.30 including postage. Playmates is *the* fanzine for all games players, includes many reviews, pokes and tips as well as Bonzo meddler news. Contact Carl Surry, 37 Fairfield Way, Barnet, Herts EN5 2BQ.

COMING UP...

To show that some forethought does go into planning each issue of Print-Out, I can tell you that the next issue of the magazine will contain all of the usual items and, in addition, the following special features:

PRINT-OUT'S DISC COPIER AND FORMATTER
- a disc copier that relieves the tedium of lots of disc swaps by using all of the additional memory which your computer has. In addition, we have a speedy disc formatting program which provides you with all the usual formats and an extra 'big disc' format.

USING MACHINE CODE CALLS FROM WITHIN BASIC
- two RSXs to allow you to pass parameters to any machine code routine, whether in ROM or RAM, from BASIC.

And coming up in future issues.....

THE EUROPEAN SCENE
- an analysis of the CPC and Plus computers in Europe by the people who really matter - the users.

SPECIAL READERS' OFFERS
- At present, we are negotiating with several leading computer software and hardware companies to bring you special discounts and some really unique opportunities in the future.

HELPLINE

RICHARD WILDEY - Help given on BASIC Programming, Tape-to-Disc transfers and the DMP2000 printer. Contact Richard at: 41 Enmore Gardens, East Sheen, London SW14 8RF.

TONY WALKER - Help given on Protext, Promerge Plus, Prospell, Utopia, Comms, CPM on ROM, ROMDOS (3.5" operating system), Star LC24-10 printers, and CPM+ Protext. Contact Tony at: 24 Ullswater Road, Fulwood, Preston, Lancashire PR2 4AT, or give him a call on (0772) 651698 (but please phone only between 10am and 10pm).

DICK HORNSBY - Help given on BASIC Programming (at ordinary level) and possibly on using Arnor's external ROMs. Help sought on more advanced BASIC programming and machine code programming. Also exchange of serious programs etc, and joint working on programs of mutual interest. You can write to me at 22 Holmwood Grove, Mill Hill, London NW7 3DT, or phone me on 081-959-4779.

If you are writing to any of these kind people, please ensure that you enclose a SAE, as you're more likely to receive a reply that way. Also please telephone only during decent hours (unless specified).

We have had a very good response to the idea of a helpline so we have printed a form for you to fill in if you're interested. Just complete it and send it off to the address at the front of the magazine.

If you want help on anything, use the same form but let us know that you are requesting help rather than offering it.

NAME (Block capitals please) .....

ADDRESS .....,

.....,

.....,

..... POSTCODE .....

TELEPHONE NUMBER .....

SUBJECTS OFFERED HELP ON .....

.....

.....

.....

.....

.....

Please tick if you do NOT want your telephone number printed in the magazine. ☐