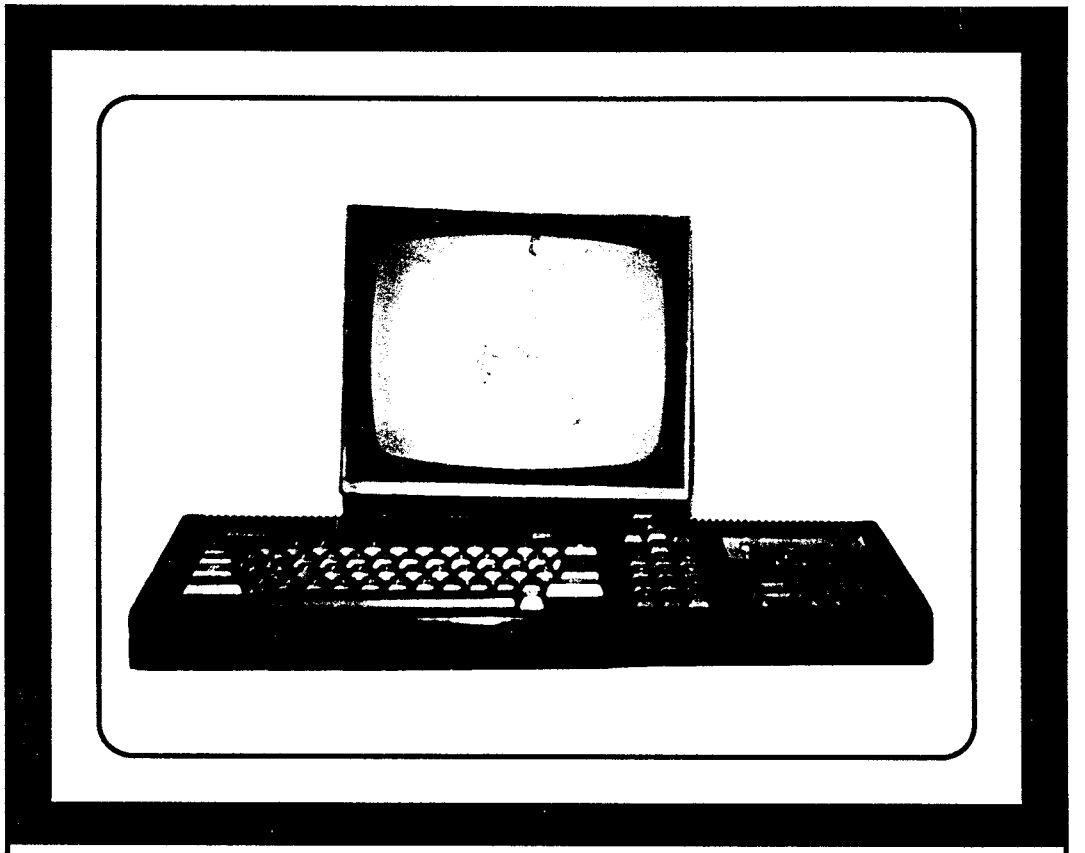
Price £1.10

PRINT-OUT

ISSUE TEN



Written by Thomas Defoe and Mark Gearing
Technical Contributors: Bob Taylor and Chris Williams

INCLUDING:

INDUSTRY COMMENT
LATEST NEWS
FIRMWARE
DISC COPIER
HELPLINE

INDEX — ISSUE TEN

Miscellaneous

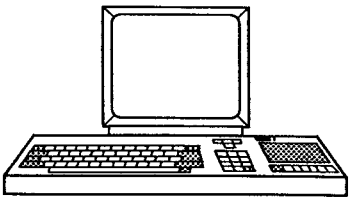
Features

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



WELCOME TO ISSUE TEN OF PRINT-OUT

It will probably amaze many of you that we have managed to get this issue of the magazine out (almost) on time - come to think of it, it amazes us! Let's just hope that this continues next issue.

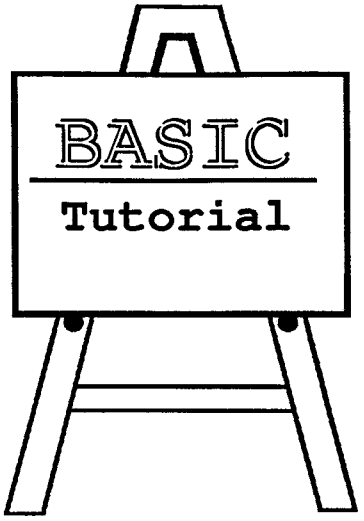
Following the excellent response from the previous issue, we are extending our offers of free copies of the magazine until 31st December. In order to get your free issue all you need to do is to introduce a friend and to get them to take out a three or six issue subscription for Print-Out - if they mention your name in their letter, we'll send you an extra copy absolutely free. Alternatively, you could write an article for Print-Out and, if it's published, you'll be entitled to a free copy. Remember, there is no limit to the number of magazines you can receive (if you are a subscriber, we'll automatically add an extra issue to the length of your subscription).

The number of people who write to ask where we advertise each issue always surprises me. However, every new issue of the magazine is announced in the Small Ads of Amstrad Action and also in Bits 'n' Pieces section of Amstrad Computer User. Other fanzines sometimes carry news of the magazine and we occassionally make it onto the news pages of the 'glossy' magazines.

You may have noticed the large increase in company advertising which comes with the magazine. Whilst I know this will annoy some people, I have tried to keep most of it out the main magazine so that no pages are taken up by the advertising. In return for this, we get free publicity which hopefully makes Print-Out more well-known - this benefits us and, in the long term, also our readers.

It is always nice to receive letters from our readers, so if you have any CPC/Plus related question, view or problem, then please get in touch with us at the address shown below and we'll do our best to answer your letter.

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



Beginner's BASIC

Computers are often used for performing many complicated functions repeatedly. One of the most common programs is the database - this allows you to enter information into your CPC, sort and tabulate it, and then at a later date to easily find the data again. There are certain rules on the kind of data that can be held in a database, but most businesses store their customers' names and addresses in a computer. Whilst this can save a lot of time, it would be pointless if you had to type in all the data every time you needed to use it. Therefore we need to have some way of storing the data in a more permanent form, such as on tape or disc.

Obviously it isn't possible just to use the standard SAVE command to transfer this data onto a disc or tape - SAVE simply records a certain part of the computer's memory (such as the area where BASIC programs are stored) and isn't meant for recording specific pieces of information. Fortunately the CPC is has a full collection of commands to do just this, and we'll be using them in this issue.

Before we can see any of these commands in action we need to take a brief look at the 'theory' behind them. If you flick through your manual you will spot many references to 'streams' - often in the context of WINDOW and PRINT. A stream can be thought of as being a means of telling the computer where to 'store' a piece of data; the CPC has streams numbered from 0 to 9. In any command, the stream's number is preceded by a hash sign (#) - eg PRINT #0

Stream 0 is the standard 'stream' and, when you switch on, it means the entire screen. If you don't specify one, then the computer assumes you mean Stream Zero and stores (ie prints) the answer on the screen. Therefore the PRINT command we often use is really an abbreviation of 'print to stream zero'.

Streams 1 to 7 are other streams which the user can define; they can be set up to cover any particular part of the screen you wish by use of the WINDOW command which we will look at next time.

Stream 8 means the printer. So, if you use a command such as PRINT #8,"Hello" the word 'Hello' will be sent to the printer and printed out.

Stream 9 refers to the tape deck or disc drive. Therefore if you type in something like PRINT #9,"Hello" then the word 'Hello' will be sent to the disc drive or tape deck. This forms the basis of the way in which information can be stored on a tape or disc. However, we still need to make sure that the data is held in the right place, and also in a form which is useful for us.

If you haven't followed everything above, then don't worry. It isn't essential for the rest of the article, although is does explain why we do somethings. Next issue we will be looking at WINDOWs and streams in much more detail.

Before we can transfer information into or out of the datafile we need to look at several new commands:

OPENIN - this OPENs a data file and tells the computer that it is to take data out of the file and bring it INto the CPC's memory

CLOSEIN - this tells the computer that you have finished bring information IN-to the CPC and wish to CLOSE this particular datafile

OPENOUT - this means that you want to OPEN a datafile and send information OUT of the computer and onto the tape or disc

CLOSEOUT - this is similar to CLOSEIN and is used when you have finished send-ing data out of the computer and onto the tape

With our first program we will send some information out to a tape or disc. so you had better ensure you've got a disc or tape ready.

```
10 REM Send information out to tape or disc
20 INPUT "Please enter your name: ",name$
30 OPENOUT "data1"
40 PRINT #9,name$
50 CLOSEOUT
```

Line 20 asks for your name and stores it in a variable 'name$'. Line 30 opens an output file called 'data1' - this means that all information sent to stream 9 (until a CLOSEOUT instruction is met) will be put in a file called 'data1'. Line 40 does exactly this by printing the contents of 'name$' into the datafile which has been opened. When this has been done line 50 tells the computer to close the output file - note that you don't need to specify a name with CLOSEOUT; this is because you can only have one file open at a time. It is very important to close both input and output files otherwise some of your data may be lost.

Having typed in and run the above program. erase it and then type in the list-ing below. When run. it will read the data created by the above program back in. and print out the result.

```
10 REM Retrieve information from tape or disc
20 OPENIN "data1"
30 INPUT #9,name$
40 CLOSEIN
50 PRINT name$
```

Line 20 tells the computer that it wants to read in some information from the file called 'data1'. The INPUT instruction is used in a similar way to the PRINT command above. and tells the computer to get 1 piece of information from a file. and then to put in the variable 'name$'. Line 40 closes the 'input channel' and line 50 prints the piece of data out - with any luck it will be the same as the information you entered using the first program. When reading back. you can use different variable names to those used when writing. So change 'name$' in lines 30 and 50 to 'info$' and try the program again - it will still work correctly.

It is also possible to send numbers to the datafile in a similar way, and also to mix the two (text and numbers) in the same file. Try the following program:

```
10 REM Send and retrieve some text and numbers
20 INPUT "Please enter your name: ",name$
30 INPUT "Please enter your age: ",age
40 OPENOUT "data2"
50 PRINT #9,name$
60 PRINT #9,age
70 CLOSEOUT
80 name$="":age=0
90 OPENIN "data2"
100 INPUT #9,name$
110 INPUT #9,age
120 CLOSEIN
130 PRINT name$;"    ";age
140 END
```

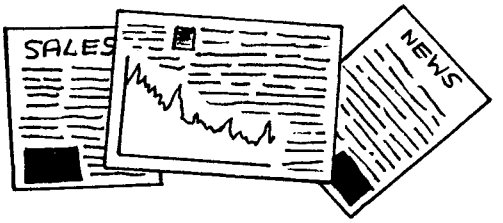
All of the program is self-explanatory. Line 80 is not necessary but is in the program to clear the variables & prove that the information is got back from the tape or disc. One thing to ensure when sending out lots of bits of data is that you retrieve them in the right order.

One important thing comes from this, and that is that you must put the pieces of data into separate PRINT #9 commands otherwise they will be stored as just a single bit of information (ie you cannot use 50 PRINT #9,name$,age) - there is a way round this and that is to use the command WRITE #9,name$,age which will work correctly. However, INPUT #9 statements can be used with multiple bits of data & so lines 100 and 110 could be replaced with 100 INPUT #9,name$,age

If you try and recall more bits of data than you placed in the datafile in the first place, the computer stops with the error EOF file met in line... EOF means 'End of File' and this occurs because the CPC came to the end of the file before it had found all the bits of data.

There is a lot more to using tapes & discs as means of storing data than we have been able to include in this issue but we're rapidly running out of room. Next time we'll look at WINDOWs in a bit more detail & also their associated commands. If we have space we will try and continue this article on using tape decks & disc drives for data storage and we'll include a program to let you read an ASCII file even without using a word-processor. Anyway, that's it for this time & I'll leave you to work out how, and why, LINE INPUT is helpful when making datafiles!!

NEWS & VIEWS



CPC Domain

PD seems to be a thriving business at the moment - not only are more libraries being set up, but some of the more established ones are entering other fields as well. CPC Domain is the latest creation of Alan Scully - it's a monthly fanzine which covers mainly PD news, although it includes a liberal sprinkling of other articles. The 42 page A5 fanzine is well produced and contains plenty of Alan's renowned humour (?). Alan seems to have an almost endless supply of contacts, so if you want to hear the latest news on the CPC scene, the chances are it will be in CPC Domain. Whilst it is mainly a 'shop window' for Scull PD's goods, it also features columns on adventuring, serious software reviews and readers' letters - littered amongst the pages you will also find many snippets of information which are essential to the CPC owner 'who has to know everything about their machine'. Of course, the rapidly growing list of the library's discs are included in CPCD along with reviews of the most popular Scull PD products. Each issue costs £1.25 (£7.50 for six issues or £15.00 for twelve) and you can get further details from Alan at: Scull PD Library, 119 Laurel Drive, East Kilbride, Glasgow G75 9JG.

Library Round-up

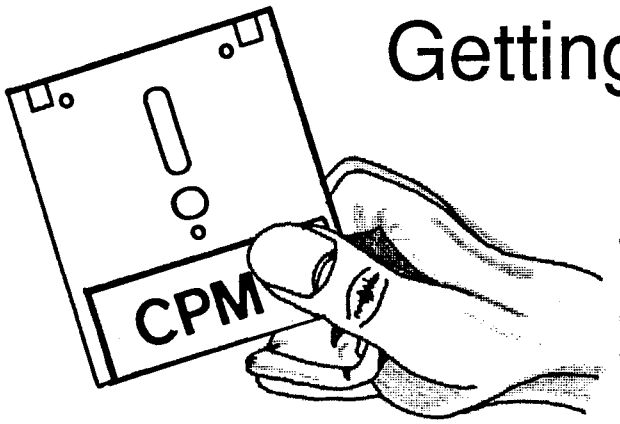
As from next issue, we will be starting up a regular section on PD libraries & their software. So if you know, or run, a library which you think we should include in our round-up then let us know. Better yet, why not review some software for this article and get a free issue of Print-Out for your troubles?

Bargains....

If anyone needed proof that the Plus computers have been highly unsuccessful, they need look no further than a recent issue of New Computer Express. In it, a company called 'Studio Software' were selling of Amstrad computers at rather low prices: CPC 464 - £100 CPC 6128 - £150 464 PLUS - £120 6128 PLUS - £200 They also do a good deal on Amstrad CF2 discs: 5 for £4.00 or 10 for £7.00. For more details their address is: Studio Software, Colne House, Highbridge Estate, Oxford Road, Uxbridge, Middlesex UB8 1UL; telephone: (0895) 272911 ext 275.

Mail Order (cont)

Following on from the mention in last issue's Editorial, we have received lots of letters from disgruntled people who have been conned by Mail Order companies. I have recently heard that one of the firms who were often criticised have begun to send out people's orders (at last). If you have a problem with a company, you should contact the local Trading Standards Authority, the magazines in which the company advertises and also the Advertising Standards Authority. If they cannot help you, and the company refuses to refund your money or send your order, have your solicitor write a letter to the company stating your position. After this, your only option will be to sue the firm through the Small Claims Court.



Getting to Grips with CPM

Introducing the system: Part 4

I would like to thank Chris Williams for taking over this tutorial, and for offering to try and help anyone with CPM - you can either write to Chris, or contact us at the usual address, and we will pass your letters on.

This is the first time I have ever written anything for a fanzine, or for anything else for that matter so I hope you will bear with me. If you find anything about this article confusing, you can always write to me at this address:

Chris Williams, 6 Frank Street, Great Horton, Bradford BD7 3BT.

Introduction...

I have had a CPC 464 for about four years and I did not really bother with CPM until I bought MAXAM II from the Home Computer club about a year ago. With MAXAM it was possible to start learning to program in CPM without having to spend days just finding my way round programs such as **ED.COM**!!! We will have to look for a more user friendly Text Editor and Assembler for CPM 2.2. If you know of one in the PUBLIC DOMAIN drop me a line and I'll pass it on.

Why use CPM?

CP/M (Control Program for Microcomputers) version 2.2 is an incredibly old OS (operating system) so why use it?

Well, you may wish to use many professional programs such as spreadsheets and databases. Most CPM utilities are too old and very hard to use, even if you can find the documentation for them.

ED.COM is so user unfriendly that the first thing you are going to have to do is find a usable text editor. You do not have to be a computer expert to use CPM (I am proof of that!), most of the programs on your CPM 2.2 disc are so old that we are going to forget all about them (!) - however we will look at some of them next time.

For example, file copying & disc management can be done with just one program instead of using two or three files on your work disc - the program that I would recommend is **NEWSWEEP.COM** and is essential to have. Space on your work disc will always be in short supply so learn to economize on the number of transient files on your work disc. Anybody requiring a copy of NEWSWEEP can send me a disc (with enough dosh for return postage) and I will send it back with a copy of NEWSWEEP on it.

We could even set up a **CPM USER BASE** if there is enough interest. In the next part we'll look at how to produce a COM file, DOC files and some more utilities from your **MASTER DISC**.

It is assumed that you have made, and are using a copy of your MASTER DISC. Never use your Master Disc as it is very expensive to replace - for details on how to make a copy of your Master Disc, read the instructions which came with CPM or are in your disc drive/6128 manual.

First BOOT UP your copy of CP/M 2.2. Erase everything except '**SETUP.COM**'; this will now be your '**WORK DISC**'. When you have finished this article, you will have your CP/M disc configured to your personal requirements, and be ready to look at some more of the utilities on your **MASTER DISC**.

Control Codes

We'll start by looking at some **CONTROL CODES** - these can be used to set up the display and colours. In addition, they can also be used for setting windows and defining characters and so on.

It's important with CPM 2.2 that whenever you change discs you must LOG it in. This is done by simply pressing CONTROL+C, this will also cause a break in most CPM programs and return you to the CCP.

The default colours (cyan) are awful and appear fuzzy on a colour TV when used with a modulator. Control Codes can be seen by pressing [CONTROL] and almost any other key. For example to set the screen mode, at the CCP press control and 'D'. A 'D' preceded by an up arrow will appear. To change the screen mode press 1 and then return. The following number can be 0,1 or 2 depending on the mode you wish to select. To change the PEN. INK and border colours, these are the codes:

CONTRL+] Followed by two parameters (first colour)(second colour).
- so "^]@@" will set the border to black

CONTRL+\@ Followed by two parameters will set the screen colours.
- so "^\@cc" will set the screen colour to red

CONTRL+\a Followed by two parameters will set the pen colours.
- so "^\azz" will set the pen to white

Each INK colour corresponds to a letter of the alphabet, for example.

CPM	AMSTRAD (colours in manual)	
@	0	BLACK
a	1	BLUE
b	2	BRIGHT BLUE
c	3	RED
d	4	MAGENTA
-	-	
y	25	PASTEL YELLOW
z	26	BRIGHT WHITE

You probably have a screen full of rubbish by now. So how do you clear the screen? This is simple, all you do is just press CONTROL+L.

A few points to note before we go onto actually running some CPM programs. Re-boot your work disc, and now type '**SAVE 0 RUN.COM**'. You will have on your disc a file called '**RUN.COM**', and this can be copied onto all your CPM 2.2 work discs. You can now load many transient programs and then change discs (don't forget to log changed discs in with CONTRL+C) and then re-run them. Run STAT.COM from your MASTER disc, now swap the master for your work disc; press CNTRL+C and type RUN. As you can see STAT.COM is re-run.

Designing A System Disc

Once you have decided which colours and mode you want, we can move on to the utility resident on your WORK DISC. It is called "**SETUP.COM**" and will be used to make your changes permanent. When SETUP.COM is run you will be presented with a list of options, most of which we will leave well alone (for now). So at the CCP type '**SETUP**'.

Option One is "INITIAL COMMAND BUFFER EMPTY, Is this correct Y/N ?". We want to leave this empty for now and we'll come back to it later. So answer 'Y'.

Option Two is "SIGN ON STRING". This is where you will input your CONTROL CODES and your sign on string. Codes such as carriage return can be built up by pressing the up (^) arrow key (shift and the pound sign), followed by the code. Below are some of codes to get you started:

Code	No of Parameters	Effect
CONTRL+D	1	SET SCREEN MODE
CONTRL+G	0	BEEP
CONTRL+H	0	BACKSPACE
CONTRL+I	0	CURSOR RIGHT
CONTRL+J	0	CURSOR DOWN
CONTRL+K	0	CURSOR UP
CONTRL+L	0	CLEAR SCREEN
CONTRL+M	0	CARRIAGE RETURN
CONTRL+P	0	SEND OUTPUT TO PRINTER
CONTRL+Y	9	DEFINE CHARACTER
CONTRL+Z	4	DEFINE WINDOW

When you've finished, answer YES to all the other questions (for now) until CPM re-boots. If you have made any mistakes, simply re-run SETUP.COM and repeat the process. You should have ended up with something like this:

^\a@@^\@zz^]@@DICKY OPERATING SYSTEMS v 2.2 1991^J^M

This string will now be printed whenever you boot up CPM, and the control codes will also be obeyed.

STAT.COM is also a useful file to have on your work disc. It allows you to set the file status & provides a sophisticated directory of the disc. Here are some of the forms of the command which you can use.

STAT *.*	
STAT filename	
STAT VAL:	
STAT DSK:	
STAT filename $r/o	sets file to read only
STAT filename $r/w	sets file to read and write
STAT DEV:	
STAT logical:=physical	assigns the specified logical device to the specified physical device.

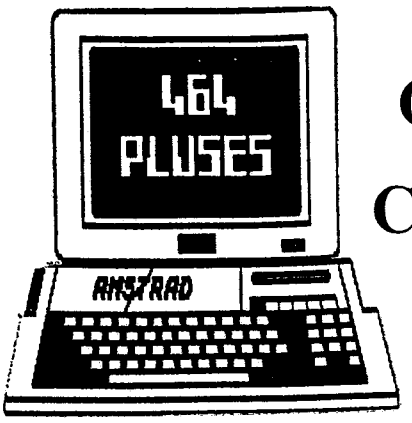
Built-in Instructions

And to end we will take a quick look at the Direct Console commands:

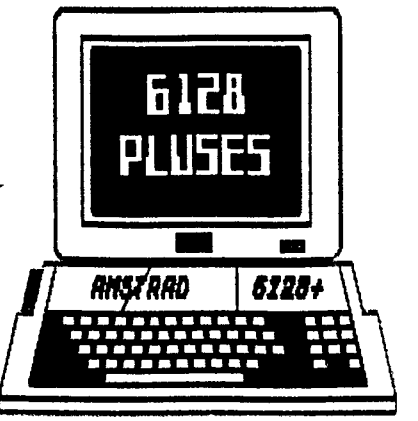
A:	selects drive A
B:	selects drive B
DIR *.BAS	lists the directory. This example will DIR all BAS files only
ERA [filename]	erases the specified file
REN [newname]=[oldname]	renames files
USER [area]	selects a user area (same as AMSDOS)
SAVE [page] [filename]	it says in the manual that this is for specialist use only, well you are about to become a specialist.

Remember that if you load a file such as STAT.COM it remains in memory until a new program is loaded. A page is 256 BYTES long and so there are 4 PAGES per 1K, [256*4=1024].

STAT.COM is 6K long, so multiply the number of K by 4 [6*4=24]. Load STAT.COM, and type '**SAVE 24 AMSTRAD.COM**'. The file AMSTRAD.COM will save to disc and it is in reality, of course, a renamed version of STAT.COM. That's it for this issue - I hope it has been some help to you. BYE !



CPC AND PLUS COMPATIBILITY



Plus 'Add-ons'

Readers of ACU may have noticed that a new PD library, called Dartsma PD, has been set up. Run by Adam Shade, this library claims to have the cheapest PD for the Amstrad CPC & Plus computers, and people can obtain a free stocklist of the discs available by sending an SSAE to him at **Dartsma PD Library, c/o Adam Shade, 47 Kidd Place, Charlton, London SE7 8HF**.

Adam runs his library on a 6128 Plus and has been so disgusted by the price of modifications for these computers, that he has produced his own range of add-ons for them. These include:

DDI-1 interface modification - allows the 464+ to properly use an Amstrad disc drive. Without alterations, the computer crashes if you reset it with the drive swtiched on, and you need to boot CPM to be able to use Amsdos.

Plus-CPC Edge Connector - unlike other designs, this plugs into the port & has a short length of cable connected to an edge connector. This stops the bulky CPC add-ons from bending the Plus' port. 464 add-ons also fit onto the Plus better.

Plus Printer Patch - this is a little box which sits between the computer and the printer lead and contains a few modifications to the wiring which stops the printer set-up from being reset when you switch the computer on or off. You will still need to buy a Plus printer lead to use a printer.

DDI-1 INTERFACE MODIFICATION	£9.00
PLUS-CPC EDGE CONNECTOR	£7.50
PLUS PRINTER PATCH	£5.50

Exclusively for Print-Out readers, Adam is offering these products at a reduced rate (shown above) or send an SSAE for more details on them. All of these prices include postage and packing.

News in Brief...

Glancing through the latest issue of Amstrad Action, I saw that WAVE have been re-advertising their own Plus computer modifications along with FD-1 second disc drives (for the 464 and 6128 Plus) at just £54.65, and also 3.5 inch disc drives (again for these computers) at a mere £85.83 - including postage and packing.

I've also heard from Carl Surry that the following games don't run properly on a 6128 Plus: Outrun, Mr Heli, Chase HQ, Midnight Resistance and also World Class Leaderboard. If you know of any other Plus compatibility problems then pass them on and we'll print them in a future issue of Print-Out.

049 &BB93 TXT GET PEN
ACTION: Gets the foreground PEN colour for the current stream
ENTRY: No entry conditions
EXIT: A contains the ink number; F is corrupt; all others are preserved

050 &BB96 TXT SET PAPER
ACTION: Sets the background PAPER colour for the current stream
ENTRY: A contains the ink number to use
EXIT: AF and HL are corrupt; all other registers are preserved
NOTES: This routine does not clear the screen when called

051 &BB99 TXT GET PAPER
ACTION: Gets the background PAPER colour for the current stream
ENTRY: No entry conditions
EXIT: A contains the ink number; F is corrupt; all others are preserved

052 &BB9C TXT INVERSE
ACTION: Swaps the current PEN and PAPER inks over for the current stream
ENTRY: No entry conditions
EXIT: AF and HL are corrupt; all other registers are preserved

053 &BB9F TXT SET BACK
ACTION: Sets the character write mode to opaque or transparent
ENTRY: For transparent mode, A is non-zero; for opaque mode, A is zero
EXIT: AF and HL are corrupt; all other registers are preserved

054 &BBA2 TXT GET BACK
ACTION: Gets the character write mode for the current stream
ENTRY: No entry conditions
EXIT: If in transparent mode, A is non-zero; in opaque mode, A is zero; always DE, HL and flags are corrupt; all others preserved

055 &BBA5 TXT GET MATRIX
ACTION: Gets the address of a character matrix
ENTRY: A contains the character whose matrix is to be found
EXIT: If it is a user-defined matrix, carry is true; if it is the lower RAM then carry is false; always HL contains address of matrix and A and flags are corrupt; all other registers preserved
NOTES: See &BBA8 (TXT SET MATRIX) for details of how the character matrix is designated

056 &BBA8 TXT SET MATRIX
ACTION: Creates a matrix for a user-defined character
ENTRY: A contains the character which is being defined; HL contains the address of the matrix to be sent
EXIT: If character is user-definable, carry is true; else carry is false; always AF, BC, DE, HL are corrupt; all others are preserved
NOTES: Matrix is stored in 8 bytes; first byte is for the top line of the character, last byte is the bottom line; Bit 7 of a byte refers to the leftmost pixel of a line, Bit 0 refers to the rightmost pixel

057 &BBAB TXT SET M TABLE
ACTION: Sets the address of a user-defined matrix table
ENTRY: DE is the first character in the table; HL is the table's address
EXIT: If there are no existing tables, Carry false, A and HL corrupt; if there was a user-defined table before, Carry true, A is the first character, HL is the table's address; always BC, DE and F corrupt

058 &BBAE TXT GET M TABLE
ACTION: Gets the address of a user-defined matrix table
ENTRY: No entry conditions
EXIT: See above (&BBAB - TXT SET M TABLE) for entry conditions

059 &BBB1 TXT GET CONTROLS
ACTION: Gets the address of the control code table
ENTRY: No entry conditions
EXIT: HL contains the address of the table; all others are preserved

060 &BBB4 TXT STR SELECT
ACTION: Selects a new VDU Text stream
ENTRY: A contains the value of the stream to change to
EXIT: A contains the previously selected stream, HL and F are corrupt; all other registers are preserved

061 &BBB7 TXT SWAP STREAMS
ACTION: Swaps the states of two stream attribute tables
ENTRY: B contains a stream number, and C contains the other stream number
EXIT: AF, BC, DE and HL are corrupt; all others are preserved
NOTES: The pen and paper inks, the window size, the cursor position, the character write and graphic character modes are all exchanged

062 &BBBA GRA INITIALISE
ACTION: Initialise the graphics VDU
ENTRY: No entry conditions
EXIT: AF, BC, DE and HL are corrupt; all others are preserved

063 &BBBD GRA RESET
ACTION: Resets the graphic VDU
ENTRY: No entry conditions
EXIT: AF, BC, DE and HL are corrupt; all others are preserved

064 &BBC0 GRA MOVE ABSOLUTE
ACTION: Move the graphics cursor to an absolute screen position
ENTRY: DE contains the user X-coordinate, HL holds the user Y-coordinate
EXIT: AF, BC, DE and HL are corrupt; all others are preserved

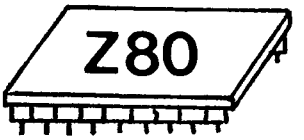
065 &BBC3 GRA MOVE RELATIVE
ACTION: Move the graphics cursor to a relative screen position
ENTRY: DE contains the X-distance to move and HL the Y-distance
EXIT: AF, BC, DE and HL are corrupt; all others are preserved

066 &BBC6 GRA ASK CURSOR
ACTION: Get the graphics cursor's current position
ENTRY: No entry conditions
EXIT: DE contains the user X-coordinate, HL holds the user Y-coordinate; AF is corrupt; all other registers are preserved

067 &BBC9 GRA SET ORIGIN
ACTION: Set the origin's position
ENTRY: DE contains the user X-coordinate, HL holds the user Y-coordinate
EXIT: AF, BC, DE and HL are corrupt; all others are preserved

Machine ...Code..

PROGRAMMING THE Z80



Division

Last issue we wrote a routine to multiply two numbers together using straightforward addition. This time we are going to look at an alternative way of doing multiplication in machine code, and also introduce a few new commands. First, we are going to see how we can perform simple division of two numbers. As with our multiplication program from last time, we are limited as to the maximum size of numbers that we can use - this is because we are just going to use the standard registers rather than trying to design our own system for storing numbers.

To simplify the explanation in this article, it will be helpful to look at the correct mathematical terms used in division of numbers. Therefore, printed below on the left is an example of division & on the right are the names for the parts of the sum.

Example: $23\overline{)367}$ = 15 + 22

Terms: divisor $\overline{)\text{dividend}}$ = quotient + remainder

Back to school

When we divide one number by another, we use a combination of multiplication & subtraction to arrive at the result. The multiplication part of the sum uses our knowledge of tables - whereas computer division ignores such tables and instead uses simple subtraction to work out the result. If we wish to divide ten by five, we can get the computer to count the number of times that five can be subtracted from ten before it reaches zero:

ie. STEP 1. 10 - 5 = 5
STEP 2. 5 - 5 = 0

and so the answer is two. It is this method which forms the basis of the simple routine to divide two numbers, which is printed below. As well as performing the actual division we will also need to make certain checks. These include ensuring that the divisor is not zero; the reason for this is that anything divided by 0 is infinity and such a result would be impossible to print. Therefore we have to spot this and print an error message informing the user of the problem.

How does it work...?

In the routine below, Register Pair HL holds the dividend and Register Pair BC holds the divisor (this means we would be able to use 16-bit numbers, providing the result could still be stored as an 8-bit number - however, for the moment we shall continue to use 8-bit divisors). Next we must make A equal to zero so that it is ready to act as a counter for the number of times we take the divisor away from the dividend. The routine then enters a loop to do this subtraction - this continues until it reaches zero: for each pass around the loop, we increase A by 1 until the dividend is zero - at this point A will hold the result.

To make the routine more flexible, it should be able to cope with numbers that don't divide exactly into each other. Because we do not know whether it will go exactly, it is impossible to detect the end of such a division by testing for 0, and so we have to test for a Carry. Such a Carry will occur when the SBC HL,BC instruction makes the dividend go below zero. The count value held in the A register will then be one too many, as the loop has gone round an extra time. Thus, A needs to be decremented by one to allow for this. Any remainder in HL is then ignored.

All that needs to happen then is for the count (which is the value in A) to be transferred into HL and this value then sent for printing by the same routine as used in the multiplication routine.

```
        ORG &8000     ; the address where we want to locate this routine
        LD HL,240     ; HL holds the value of the number to be divided (dividend)
        LD BC,60      ; BC holds the value of the number to divide by (divisor)
        LD A,B
        OR C          ; check to see if the divisor is zero
        JP Z,prterr   ; if it is jump to the error message printing routine
        LD A,0        ; A = 0 ie set the counter to zero to record the number of
                      ;  times that HL can be divided by BC
        OR A          ; clear the Carry flag for the SBC HL,BC which follows
.loop   INC A         ; increment the counter by one
        SBC HL,BC     ; subtract the divisor once each pass round this loop
        JR NC,loop    ; and repeat if the dividend has not yet become negative
        DEC A         ; when it does go negative, we must adjust the counter for
                      ;  going once too many
        LD H,0        ; this and the next instruction load the result (quotient)
        LD L,A        ; from the A register into the HL register pair
        OR A          ; clear the Carry flag for during .print routine
        CALL print    ; call the routine to print the value
        RET           ; and finally return to BASIC

.prterr LD HL,errmes ; load HL with the address where the error message is
        CALL prtstr  ; now call the routine which prints a string of text
        RET          ; return to BASIC and abandon the calculation

.prtstr LD A,(HL)
        CP 0
        RET Z
        CALL &BB5A
        INC HL
        JP prtstr

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

(continued in next column)

(continued from previous column)

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

.errmes DEFB 10,13,"Overflow: division by 0",0
```

The last three routines have all been printed in previous issues of Print-Out together with detailed explanation of how they work, and so we won't bother going into them again in this article. To use any of these routines in this machine code article you'll need an assembler such as the one on the program disc/tape from Issue Seven of Print-Out.

Bit shifting

Last issue we performed multiplication by adding numbers together, but it can also be done using more complex, and flexible, routines which rely on being able to 'rotate' registers. You may already have seen the trick of doubling a binary number by just shifting its digits along to the left and then adding a 0 to the end:

eg 1010 (binary) = 10 (decimal)
10100 (binary) = 20 (decimal)

So much for multiplying a number by 2 (ie doubling it), but what happens if we want to multiply 1010 (binary) by three? All we do now is multiply the number by two and then add the original number to the answer:

eg. 1010 (binary) times 3 equals 1010 (binary) times 2 plus 1010 (binary)

With a little extra thought, it ought to be apparent that using this process you could multiply together any numbers by using this combination of shifting a binary number's digits to the left and then adding itself in - this forms the basis for multiplication.

Unfortunately, we haven't got enough space in this issue to go into all of the shift and rotate instructions - so for the moment you will have to take some of the commands on trust - however, we'll try & cover them fully in our next issue.

In practice, the HL register pair is assigned to hold the result of the multiplication and the other two register pairs hold the two numbers to be multiplied together. HL is set to zero. Each bit of the multiplier (BC) is tested in turn, by rotating its register pair a binary digit at a time to the right and checking to see if the Carry flag is set. If it is, the current value of the multiplicand (DE) is added to HL; if it isn't, then no addition takes place.

Between each stage of checking & adding in (or not), DE is rotated to the left (ie making it two times larger). These rotations are repeated 8 times in all, so that all of the digits of the multiplier are checked.

We also have to ensure that all of the values generated in the program are not so big that they cannot be stored in their relevant register pairs. Bearing this limitation in mind, we have the nucleus of a machine code multiplication routine and it would appear like this:

```
.multip LD DE,multiplicand
        LD BC,multiplier
        LD HL,0         ; set the 'product' to zero
        LD A,8          ; 8 passes to make for an 8 bit multiplier
.loop   SRL B           ; rotate the HB of the multiplier (not really necessary if
                        ;  the multiplier is always limited to 8 bits)
        RR C            ; rotate its LB also - its lowest bit will go to the Carry
        JR NC,noadd     ; test the Carry for a bit; don't 'add in' if no bit
        ADD HL,DE       ;  otherwise add in the current value of the multiplicand
.noadd  SLA E           ; double the size -
        RL D            ;  of the multiplicand
        DEC A           ; check whether 8 bits have been done
        JR NZ,loop      ; if more to do
        OR A            ; remove Carry for .print
        CALL print
```

This isn't a complete program - you need to fill in values for DE and BC, and to add a printing routine at the end. If you can't follow what is going on in this routine, don't worry as we'll look at it again next time.

If you have any questions, information or just have something you want to say, we would be delighted to hear from you.

ROM Blowing

I was pleased to see that your magazine is gaining strength. All the best with the service. My present interest is in writing a ROM with Utopia type facilities which work with a 3.5" B drive. I am only starting out on this venture so would be interested in being put in touch with anyone with ROM blowing or file copying experience. I would also be willing to share what experience I do have so please put my details into your magazine:

SAM WRIGHT, 24 Chester Avenue, Whitehead, Carrickfergus, Co. Antrim BT38 9QQ.

PRINT-OUT: You might find the articles on RSXs in Issues Four to Eight helpful - Bob Taylor is familiar with blowing ROMs, and has several home-made ROMs which he swears by!! If anyone has any further information, then either write to Sam or get in touch with us and we'll pass it on.

All Sewn Up

Having read Mrs Brockbank's letter on 'Knitting on the CPC'. I have some info. There are some programmes available for the CPC (Spectrum, C64 and Spectrum). I, or I should say my wife uses them; they are very good & save a lot of time. they suit both hand and machine knitting as follows:

1. Classic Styles (round necks, 'V' necks, raglans, etc)
2. Fashion Tops & Batwing (drop shoulders and sideways knit)
3. Tops 2 & Cardigan (more advanced styles)
4. Skirts Vol 1 (variety of downward knitting skirts and dresses)

Prices are £9.95 each (tape) or all four for £34.95 on disc. They are available from: Terry Mason, 15 Inishmoyne Green, Antrim, Northern Ireland, BT41 4JZ.

PHILL MACKAY
BARRY

Reading through Issue Nine, I saw Mrs Brockbank requesting a knitting/sewing program. Well, in issue number 24 of Amstrad Action a type-in was printed called 'WEAVING'. I am not sure whether it would be useful in this circumstance but it is rather interesting.

ADRIAN SILL
DONCASTER

PRINT-OUT: Thanks for the information which I'm sure will be appreciated by all knitters out there. It just goes to prove that our readers can probably answer any CPC problem we throw at them!!

8-Bit.....R.I.P.

I am writing this letter with a great sadness in my heart. The reason is very simple - I've just bought a new PC386 which now holds pride of place in my home. I don't think that I can ever go back to playing on my green screen CPC464 after seeing some of the amazing graphical screens and the speed of my new PC. Do not get me wrong, I have been very loyal to Amstrad ever since Day One. I bought my CPC464 in October 1984 and around 1986 I bought an Amstrad PC1512 - what a mistake that eventually turned out to be!

But back to my new PC: the much better graphics (VGA) are a treat to look at, especially Space Quest IV (I would love to see that on a CPC) which is stunning. With an 'adlib' card in the system, the whole set-up is just amazing. Admittedly the price of a brand new PC is very expensive (mine cost me an overall total of £1400) but so far, and I've only owned it for a month, I can't believe that I've ever lived without it. Ah! I hear you cry but PC games are so dear (£30-£40). If you order by mail order and know that you like the game, it's well worth all the money because of the amount of work that has gone into the program.

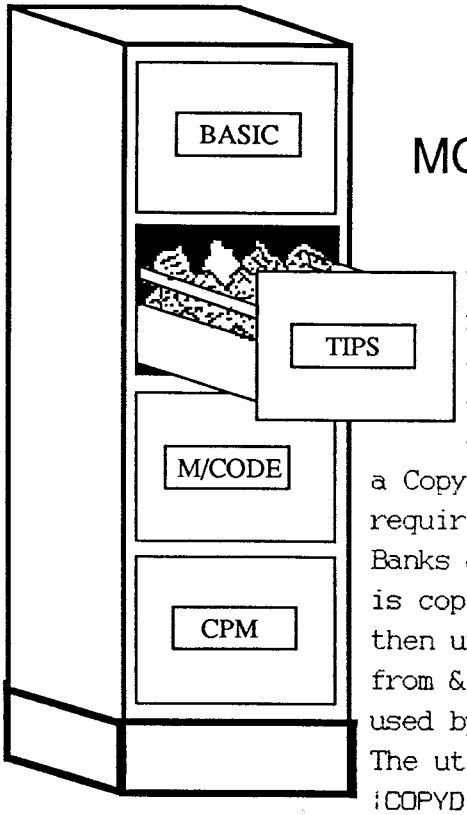
I think that it is time that we admitted that 8-bits have come and gone - you need only to look at sales (US Gold putting two different formats of games onto one tape, so as to save money) and the amount of magazines that are now around - in 1985 there were about six or seven different magazines for the Amstrad; these have either collapsed, left the CPC market behind, or are just getting worse.

You will probably tell me the exact opposite and the CPC market is still booming, but for how long? I'm sorry to have to say this but the 8-bit market is now dead - take the GX4000 console, Dixons can't sell them & I heard they were being flogged for £30. The CPC is not fast enough or powerful enough to come upto the standards for today's computers and software - after all, there is only so much that a programmer can do with an 8-bit.

ALASTAIR HENDERSON
OCKHAM

PRINT-OUT: Whilst I hate to agree with you, it seems as if you are not the only person to have lost faith in the CPC. Indeed, recently Amstrad have cancelled plans to produce new disc drives and accessories for the Pluses. Instead, they are devoting their efforts to promoting a new PC-compatible games machine that they hope will compete with the Amiga and Atari ST - wishful thinking! Readers of Future Publishing's 'New Computer Express' will have noticed that the magazine has all but dropped its coverage of the 8-bit machines and it is instead concentrating on the lucrative 16-bit market.

Having said all of that, it's important to remember that people want different things from their computer. Whilst you argue that £30 is a reasonable price to pay for a game, providing your know that you like it, it seems highly unlikely that everyone will be prepared to shell out that kind of money - especially if you consider that many home computer users are children. Many people value the CPC because they can actually program it (try 'true' programming on a PC) and can run serious applications (most business programs cost £150 to £500). Even though the CPCs days may be numbered, I can't honestly see the PC as being its natural successor - besides, how long will it be before 16-bits are classed as being 'out-dated' ?



Technical Tips

MORE READERS QUERIES ANSWERED.....

by BOB TAYLOR

The utility described here was written in response to a recent request. It has the limitations that it requires a 6128, or a 464 fitted with a 6128 ROM plus a 64k RAM expansion, which can be supplemented with a 256k memory expansion on either machine. If 256k RAM pack is fitted, a Copy or Verify procedure can be achieved using a single pass requiring just one disc swap and this is acheived by using RAM Banks &C4 to &D7. With less than this amount of RAM, the disc is copied in two passes, needing three disc swaps. The routine then utilises RAM Banks &C4 to &C7 plus the area of normal RAM from &1800 to &A000, so make sure that this area is not being used by a BASIC program, or sideways ROM.

The utility provides two new RSXs. The syntax for Copying is: |COPYDISC, and to verify a disc use: |VERIFY

You will be prompted to insert the SOURCE disc and the disc to COPY TO. If the second pass is required, you will be prompted for these discs again. At any time in |COPYDISC or |VERIFY, [ESC] can be pressed to abort the process.

The disc to copy to does not need to be formatted, formatting (or reformatting of a disc with a different format to the SOURCE disc) takes place as each track is written. SOURCE discs must always be formatted, and discs to Verify must have the same format as the SOURCE disc - that is obviously essential, otherwise the disc can't be a copy anyway).

When Verifying, you will again be prompted for the SOURCE disc and the disc to Verify, as often as they are needed. If a Verify procedure requires two passes, this can be done with only two changes of disc - ie from the SOURCE disc to the disc to Verify, which can be used as the 'SOURCE disc' at its second prompt, and then back to the SOURCE disc inserted again as the second 'disc to VERIFY'.

A further facility can be chosen by using: |VERIFY,anything

This is intended to verify more than one disc against a SOURCE disc which has already been loaded into memory, and so really requires the 256k RAM expansion. With only the smaller expansion fitted the service is still available, although only half a disc can be verified at a time. This will usually be the second half (tracks 20 to 39 for DATA or SYSTEM formats), since this will be the contents of the RAM Banks after completing a full Copy or Verify.

However, any number of first halves can be checked by Verifying the first half of one disc (using |VERIFY only), and then pressing ESC at the second prompt for the SOURCE disc. ESC will also need to be used after each half disc is verified, to stop loading in the second half of the SOURCE disc.

Both |COPYDISC and |VERIFY load tracks into the same Banks of RAM. so a Verify command can follow either of these commands.

Sectors which don't match are printed on the screen. and since this may take a little time if a completely unmatching disc is inserted. a Verify can be aborted at any time by using ESC.

Type in the BASIC listing below and SAVE it before RUNning it. This will provide a block of code to be used straight away. or re-LOADed later. Instructions for using the code are given in the BASIC program. which prints them at the time of running. These should be copied for subsequent use.

* Remember when you copy to a disc it loses all the files it held previously! *

PROGRAM:

```
[F1]  10 'DISC COPIER and VERIFIER Loader  by Bob Taylor  for 6128 ROM only
         (copyright 1991)
[D5]  20 MEMORY &97FF:RESTORE:PRINT:PRINT"Please wait a few seconds"
[2D]  30 FOR lin=0 TO &300/8-1:total=0:FOR n=0 TO 7:READ a$
[9D]  40 byte=VAL("&"+a$):POKE &9800+lin*8+n,byte
[4B]  50 total=total+byte:NEXT n
[0D]  60 READ a$:IF VAL("&"+a$)<>total THEN PRINT:PRINT"Error in line"lin*10+110
         :PRINT:END
[C4]  70 NEXT lin
[D1]  80 PRINT:PRINT"All M/C loaded":PRINT:PRINT"Press S to save M/C as
         COPYDISC.BIN":PRINT"or any other key to continue":WHILE INKEY$="":WEND:
         IF INKEY(60)<>-1 THEN SAVE "COPYDISC.BIN",B,&9800,&300
[2D]  90 PRINT:PRINT"To Load and Initialise |COPYDISC and |VERIFY RSXs just
         use:":PRINT"MEMORY &97FF:LOAD"CHR$(34)"COPYDISC.BIN"CHR$(34)":CALL
         &9800":PRINT"with the Disc or Tape inserted"
[EA] 100 END
[D6] 110 DATA EB,36,C9,23,01,0B,98,C3,374
[43] 120 DATA D1,BC,01,00,9A,F6,80,06,3A4
[9D] 130 DATA AF,32,FB,9A,E6,01,3E,FF,49A
[F1] 140 DATA 32,FC,9A,20,50,57,3E,01,2CE
[A0] 150 DATA 32,66,BE,32,78,BE,06,C3,387
[1F] 160 DATA 21,14,9A,1E,00,22,FD,9A,2A6
[D6] 170 DATA 2A,FD,9A,CD,01,99,CD,06,3FB
[35] 180 DATA BB,FE,FC,CA,E1,98,DF,EA,6C1
[37] 190 DATA 9A,30,ED,3A,51,BE,E6,F0,4D6
[5C] 200 DATA C6,09,4F,21,00,7E,58,ED,302
[A9] 210 DATA 53,F9,9A,5D,CD,A1,99,CD,517
[5B] 220 DATA 83,99,28,07,DF,F0,9A,38,3EC
[03] 230 DATA F3,18,78,21,38,9A,3A,FB,3AB
[4D] 240 DATA 9A,CB,7F,28,03,21,61,9A,32B
[76] 250 DATA CD,01,99,CD,06,BB,FE,FC,4EF
[D9] 260 DATA 28,67,AF,DF,ED,9A,38,02,3DE
[48] 270 DATA 23,7E,CB,6F,28,DD,DD,21,3DE
[33] 280 DATA CF,9A,3A,51,BE,E6,F0,28,4B0
[B3] 290 DATA 0B,01,09,00,DD,09,FE,40,239
[5F] 300 DATA 28,02,DD,09,81,4F,ED,5B,328
[52] 310 DATA F9,9A,43,21,00,7E,5D,3A,30C
[72] 320 DATA FB,9A,CB,7F,20,5F,CD,A1,4CC
[46] 330 DATA 99,CD,83,99,28,3D,38,1E,33D
[34] 340 DATA C5,E5,DD,E5,C1,1E,09,21,475
[CB] 350 DATA 21,9B,2B,36,02,2B,0A,77,1CB
[93] 360 DATA 2B,36,00,2B,72,03,1D,20,13E
[9A] 370 DATA F1,DF,F3,9A,E1,C1,DF,F6,6D4
[64] 380 DATA 9A,38,D3,21,89,9A,CD,01,3B7
[A3] 390 DATA 99,21,C2,9A,CD,01,99,3E,3BB
[F3] 400 DATA 10,32,66,BE,AF,32,78,BE,37D
```

(cont. over)

```
[3C] 410 DATA C3,5B,BD,7A,FE,28,28,EF,492
[1D] 420 DATA 15,06,C3,21,0F,9A,C3,2B,296
[57] 430 DATA 98,7E,B7,C8,CD,5A,BB,23,49A
[1C] 440 DATA 18,F7,E1,18,CE,CD,A1,99,4DD
[7C] 450 DATA CD,83,99,28,DE,D5,E5,26,4CF
[A3] 460 DATA 9C,DF,F0,9A,EB,E1,30,EA,5EB
[D3] 470 DATA 1A,BE,28,4C,D1,D5,E5,3A,411
[BE] 480 DATA FC,9A,BA,3E,2C,28,1E,21,321
[BE] 490 DATA A5,9A,CD,01,99,7A,32,FC,44E
[94] 500 DATA 9A,1E,2F,1C,D6,0A,30,FB,30E
[1C] 510 DATA 57,7B,FE,30,C4,5A,BB,7A,453
[FF] 520 DATA C6,3A,CD,04,99,CD,5A,BB,44C
[B0] 530 DATA 3E,43,CB,79,20,08,3E,34,25F
[E5] 540 DATA CB,71,20,02,3E,30,CD,5A,2F3
[B4] 550 DATA BB,79,E6,0F,C6,30,CD,5A,446
[AB] 560 DATA BB,11,FF,01,E1,CB,C4,6B,4A7
[EF] 570 DATA 23,13,CB,4A,28,AA,25,25,267
[B5] 580 DATA D1,CD,09,BB,FE,FC,20,8D,509
[DF] 590 DATA C3,E1,98,0C,79,E6,0F,FE,4B4
[4F] 600 DATA 08,D8,FE,0A,79,30,04,E6,37B
[5F] 610 DATA F0,37,C0,E6,F0,3C,4F,14,45C
[31] 620 DATA 3E,28,CB,7C,28,01,7A,BA,30A
[19] 630 DATA C9,24,24,CB,7C,C8,CB,78,463
[43] 640 DATA C8,26,40,D5,04,CB,D0,78,41A
[E4] 650 DATA CD.5B,BD.56.78,FE.CC,38,4B5
[9F] 660 DATA 2A,36,FF,E6,C7,CD,5B,BD,4F1
[F9] 670 DATA 5E,34,20,19,34,CD,5B,BD,2E4
[F2] 680 DATA 35,72,20,0E,78,FE,CC,20,337
[41] 690 DATA 27,D1,26,18,CB,B8,AF,C3,42B
[45] 700 DATA 5B,BD,CD,5B,BD,73,CD,5B,498
[91] 710 DATA BD,72,56,3E,AA,77,BE,20,3C2
[44] 720 DATA 03,2F,77,BE,72,20,09,23,225
[E3] 730 DATA CB,74,20,EE,D1,26,40,C9,44D
[18] 740 DATA D1,F1,21,95,9A,C3,DE,98,54B
[41] 750 DATA 56,45,52,49,46,D9,43,4F,2E7
[A9] 760 DATA 50,59,44,49,53,C3,00,0D,259
[2A] 770 DATA 0A,72,65,2D,69,6E,73,65,2BD
[C9] 780 DATA 72.74,20,53,4F,55,52,43,292
[2A] 790 DATA 45,20,64,69,73,63,20,2D,255
[01] 800 DATA 20,74,68,65,6E,20,61,6E,2BE
[D2] 810 DATA 79,20,6B,65,79,0A,0D,00,1F9
[09] 820 DATA 0A,69,6E,73,65,72,74,20,2BF
[5D] 830 DATA 64,69,73,63,20,74,6F,20,2C6
[BA] 840 DATA 43,6F,70,79,20,54,6F,20,29E
[3A] 850 DATA 2D,20,74,68,65,6E,20,61,27D
[07] 860 DATA 6E,79,20,6B,65,79,0A,0D,267
[26] 870 DATA 00,0A,69,6E,73,65,72,74,29F
[9C] 880 DATA 20.64.69.73.63,20,74,6F,2C6
[ED] 890 DATA 20,56,65,72,69,66,79,20,2B5
[31] 900 DATA 2D,20,74,68,65,6E,20,61,27D
[FE] 910 DATA 6E,79,20,6B,65,79,0A,0D,267
[1B] 920 DATA 00,0A,44,69,73,63,20,66,213
[B8] 930 DATA 61,75,6C,74,00,0A,52,41,253
[36] 940 DATA 4D,20,42,61,6E,6B,20,66,26F
[A1] 950 DATA 61,75,6C,74,00,0D,0A,45,212
[18] 960 DATA 72,72,6F,72,20,69,6E,20,2DC
[3F] 970 DATA 54,72,61,63,6B,20,00,2C,241
[C7] 980 DATA 20,53,65,63,74,6F,72,20,2B0
[62] 990 DATA 26,00,3A,20,41,62,6F,72,204
[7C] 1000 DATA 74,65,64,0A,0A,0D,00,00,15E
[A4] 1010 DATA 08,07,06,05,04,03,02,01,024
[76] 1020 DATA 45,49,44,48,43,47,42,46,22C
[69] 1030 DATA 41,C5,C9,C4,C8,C3,C7,C2,5A7
[00] 1040 DATA C6,C1,6C,C5,07,30,C6,07,3BC
[B3] 1050 DATA 66,C6,07,52,C6,07,4E,C6,366
[82] 1060 DATA 07,00,00,00,00,00,00,00,007
```

RAMROM review

At last it's come! I've finally received a RAMROM from Microstyle after almost a year of waiting for my order to be sent. Now what you may ask, is a RAMROM and why have I been patiently waiting that long without demanding my money back?

To answer the last question, I have tried to obtain a refund several times but Microstyle aren't very 'user-friendly' about assuring customers that their order is still safe, nor forthcoming with refunds if there is a long wait.

So what does a RAMROM do? Programs such as PROTEXT and MAXAM are available in different media forms - namely Cassette, Disc and ROM. The first two types have to be loaded in each time the computer is reset or switched on, but ROM programs are ready to use immediately without any loading. A ROM is just a block of fixed memory which is switched into the CPC's memory whenever it is needed.

In fact, this memory does not have to be fixed (that is, in ROM form), and so it's possible to use RAM instead, with the advantage that programs can be easily altered. However, such a program (your own of course) in RAM will disappear when the CPC is switched off, and so will have to be loaded in at switch-on. All this doesn't sound much better than disc programs as far as loading goes, but now the advantages begin to appear. These 'ROM' programs will survive a reset or a crash which doesn't require switching the computer off; also they don't take up any of the normal RAM space (except for a small area which they may reserve for a work space).

The standard RAM which comes with the computer cannot be used for a RAM-ROM as it's connected differently, and the same applies to the Banked RAM. What we need is RAM which can be switched in using the appropriate Ports and that is what the RAMROM provides.

It comes on a small PCB having three switches, a through connector so that any other items can be plugged after it, and also four chips, one of which is a 32K RAM chip allowing it to simulate two ROMs. These two blocks of 16K can be set to behave like two of ROMs 3, 4, 5 and 6 (in a RAMROM certain combinations of these four ROM numbers are possible) or alternatively, in a parallel mode of operation whereby each has the same ROM number (only 4 or 6 in this form), but only one is in use at a time.

This parallel feature allows an extra 'ROM' to be available without taking an extra ROM number, but it does mean that both 'ROMS' can't be available together, and that if the other one is selected, it will need to be logged on by resetting the computer (this requirement can be ignored if the Initialisation routines are very similar).

The device works as intended, which doesn't sound much, but the convenience of being able to load machine code programs into ROM to try them out and then being able to alter them just as easily (without all the fuss of blowing a proper ROM, then erasing it and re-blowing it to accommodate any changes) can only be understood by someone who has had to use that development path!! A piece of hardware like this allows even the novice machine code programmer to get into ROMs.

Coming back to losing the RAMROM programs every time the computer is switched off, it's possible to add a few components to the board to allow the programs in the ROM 4 or 6 position to be retained after switch-off.

All in all, I can reccommend the RAMROM as a very useful and inexpensive piece of kit at only £13.95, with the proviso that you should enquire about its availability before ordering (some of our readers have also expressed difficulty when ordering from this company - Ed).

SOUND

An introduction to the CPC's sound chip
PART 3

Apologies for the shortness of this issue's aritcle - there is just so much to squeeze in - but hopefully we will be back to our usual size in the next issue.

Last time we introduced the more complicated Volume Envelope and programmed in a simple tune, which also utilised an envelope to give it more depth. However, a volume envelope isn't the only way of changing the sound of a note - we can also change its pitch, or tone, by use of the ENT command.

The ENT command defines an envelope which controls the actual 'note' which the computer is playing. Unlike volume changes, it's harder to notice slight variations in pitch when listening to sounds in real life. They do exist, however, and are especially noticeable in musical effects such as vibrato and trills, as well as in sound effects.

Just as the ENV command varied the volume of a sound, so ENT changes the tone of the sound - that is it makes the note become higher or lower. Here's the full command syntax (don't be put off by its length as it's just like ENV):

ENT S,T1,V1,W1,T2,V2,W2,T3,V3,W3,T4,V4,W4,T5,V5,W5

This can be broken down into five sections which can be looked at independently. 'S' can be a number from between 1 and 15 and lets the CPC know which tone envelope we wish to set-up. It's useful to know that if this number is negative the computer will continue to repeat the envelope until the end of the note has been reached - this is extremely useful when playing musical tunes.

Each of the five sections controls the tone of the note for a given length of time and then hands over control of the note to the next section. As with volume envelopes, you don't have to use all five sections but each section used must be complete (ie have three parameters).

T is the STEP COUNT and has the same meaning as for the volume envelope. It has a range of between 0 and 239: if T equals zero then it acts like a value of 1.

V is the STEP SIZE and again means the same as for the volume envelope, & it can be in the range of between -128 and 127. Note, that if the pitch goes outside the range allowed (0 to 4095) then it 'folds over' as happened with the volume envelope. If V equals zero, then this section will hold the pitch constant for a duration equal to T*W.

W is the PAUSE TIME & also is the same as for the ENV command and can range from 0 to 255. If W equals zero then this is treated as if W equals 256.

It is important to remember that the tone envelope has absolutely no effect upon the length of the note - when a tone envelope 'runs out', control is just handed back to the simple SOUND command.

LATE NEWS

Since the editorial and news pages of this issue were written we have received several pieces of important news for CPC and Plus owners.

New Products Released

SD Microsystems have now released the two products which were mentioned in the interview with Steve Denson. Their 3.5" second disc drive package comes complete with a high-capacity formatter and five blank discs. It uses the TEAC mechanism which is widely regarded as the best drive available. and the software supplied for using the drive's 800K capacity includes a disc manager & a 3" to 3.5" file copier. You can select whether the drive is to act as a standard 180K B drive or as a 396K per side drive. The complete package costs £79.95 plus £4.50 post and packing, although Print-Out readers can order the package post-free.

The second product is an art package called Picasso. and includes advanced art tools and features - such as curves. spherical fills, pattern fills and even a perspective fill for 3-D effects. It contains a machine code printer dump which allows you to print out screens in a choice of four sizes. It costs £14.95 + £1 postage and packing on disc only. and we hope to bring you more details on these products in our next issue.

Scull PD Changes

Scull PD. the highly successful PD library run by Alan Scully. has changed its PD distribution service. Discs are now loaned to you with the requested software on. and it is upto you to copy the discs: if they're not returned within a week. there are some hefty fines and penalties to pay. To order. you need to send the following: a sticky self addressed label, a 24p stamp, your name & address. the name of the disc(s) you want. and £1.50 (or 75p if you subscribe to CPC Domain). Scull PD's address is: 119 Laurel Drive. East Kilbride. Glasgow G75 9JG.

Disc Shortage

Amstrad have now stopped producing 3 inch disc drives and, more significantly. three inch discs - so no more Amsoft discs. This is bound to push prices up. so it is a good idea to buy a large number of discs now while the prices are still relatively low.

NCE is dead

The last issue of New Computer Express. the weekly computer magazine. has just rolled off the Future Publishing presses. Whilst its 8-bit section has declined over the past months, many people enjoyed its 'across the format' approach which let them keep up-to-date with all the latest developments in the computer world. Ironically. New Computer Express' blames its demise on the success of Future's other monthly magazines. including Amstrad Action.

We have recently updated the program tape and disc from Issue Nine to include several extra programs that were not in the magazine. These three programs allow you to back up valuable (or rarely used) discs on to tape and then restore them when required. The routines are very versatile and will handle almost any files: unlike most programs of this sort. you can still load individual BASIC & binary files from the tape as normal. or you can choose to restore the entire tape. All processes are virtually automatic and require the minimum of attention from the user.Issue Ten's program tape/disc includes a listing to produce cassette labels quickly and with the minimum of fuss.

We have also produced a new front-end for our recent discs and this allows you to see at a glance what programs are on the disc. get information on them. view helpfiles and even run them.

In the CPM tutorial a program called 'NEWSWEEP' was mentioned - it might be of interest to know that Scull PD can supply this program. along with several other programs. on a disc entitled CPM1 - Utilities/Applications.

Since we changed to producing the magazine on A3 paper. we have been indebted to those people who have spent many hours folding all of the sheets. and then assembling Print-Out. Special mention goes to William Defoe for all his work.

PLAY MATES

Run by Carl Surry

Play Mates is a fanzine for people who like to play games on their CPC. It has reviews of games which are written by the readers of Play Mates. Also there are the odd page or two of games tips and pokes.

One of the main sections of Play Mates is called the "Bonzo Litter Tray". This section is for the users of the Bonzo Super Meddler and Bonzo Blitz discs. These discs will transfer over 1000 tape games to disc (Please note that multi-load games will only transfer the main file, levels will still have to be loaded from tape) for faster loading and ease of use. In the Bonzo Litter Tray section I pass on any news of new transfers or sometimes a new type-in to get a game to disc. Plus we sometimes get new games loaders that have pokes in them so you can load and poke the game at the same time.

How if you fancy a copy of Play Mates and want to know how to get yourself a copy read on. Play Mates comes out every 3 months, on the 1st of March, June, Sept and Dec. It costs £1.30 an issue, this price includes postage. Send your orders to :

Carl Surry

37 Fairfield Way, Barnet, Herts, EN5 2BQ

Now Play Mates has been called one of the best CPC fanzines around. That may or may not be true, I leave that for you to judge. But I will say this, it is only as good as it is because of the contributions from its readers. They write most if not all the reviews, pass on pokes and games tips, plus of course any Bonzo news. So I would just like to thank them and you, if you fancy being a reader of Play Mates and fancy your hand at reviewing a game and wish to see it printed in Play Mates.

PLAY MATES

For CPC games reviews

PLUS BONZO NEWS

Bonzo News

Do you want to transfer your tape games to disc ? If you do try the Bonzo discs from Microstyle. Here are just a couple of well known games that will transfer:-

Welltris via Option 1
Silkworm via Blitz SX

THIS BONZO HAS BEEN PUT TOGETHER USING PAGEMAKER ON AN AMSTRAD PCW8256 FROM HEAD SCULL

SOFTWARE REVIEWS

PROGRAMMING

This issue, we are looking at some of the software available for programmers, including Multi-Code RSX from SD Microsystems, Sim from Goldmark & Sprites Alive by Glenco. But do they actually help you program...

Multi-Code from SD Microsystems (Price: £12.50 on disc)

Multi-Code RSX is a set of extension instructions which can be used from BASIC by using bar (|) commands. Of the 56 RSXs in the relocatable toolkit, most deal with manipulating the screen. In addition there are the obligatory DEEK and DOKE instructions which are two-byte versions of PEEK and POKE; a key repeat setter; a command to swap the contents of two variables (INTeger, REAL or String); and a handy RSX to display Programs with each statement on a new line.

Included in the graphics commands are scrolling routines, of which there are 2 varieties - character scrolling that is fast but jerky, and also pixel scrolling which is slow but more versatile. There are several commands which allow you to clear the screen in different ways - by 'rolling' the screen upwards or sideways or by fading the screen out. There are also commands to invert the display, dump text to the printer, copy an area of the screen, draw boxes and fill them.

Some of the more useful features included a menu template creator, a sideways string printing routine (which was quite neat), double height letter printing, plus a number of commands to give a 464 some of the extra features of a 6128.

There are also 31 RSXs supplied on the disc for control of the printer - they are mainly substitutes for standard control codes for operating its features and type faces. The commands include ones to set NLQ, Condensed, Bold, Superscript, page margins plus some to feed the paper or move the print head - all of these can easily be done from BASIC or from most wordprocessors. However, this section did include an RSX to check whether the printer is on and ready which is really quite useful.

Demo programs for both sets of RSXs are included on the disc; the printer demo crashed when run (this can be solved by changing line 90 of the BASIC program to read 'CALL HIMEM+71') but even then it tries to use ten unknown commands.

Whilst there is a large selection of commands available in this package, many can be achieved through a bit of careful BASIC programming. Looking through the 9 page A5 instruction booklet, the RSXs seem to have a large number of drawbacks and conditions when they must not be used. For me, a toolbox really needs to be almost invisible to the user and most of these problems should have been ironed out of the code.

Providing you read the manual carefully and observe the conditions laid down, all the commands are competent and do their job well. However, there is nothing incredibly original on the disc and it all depends on whether you're prepared to sit down and try to write your own routines or not.

Sprites Alive

from Glenco Software (£22.95 on disc only)
(£29.95 with compiler on disc only)

This is a suite of utilities for producing Sprite based games; such games are primarily intended to be run from BASIC & to this end, the core of this product is a set of 70 RSXs. In order to create Sprites to go into the games, a Designer is provided. An optional extra is a Compiler which produces stand alone code for much faster games.

The commands are quite extensive, covering most aspects of operating sprites, and include: placing and removing sprites, setting what will happen when it gets to the edge of the screen or when it hits a piece of scenery, moving the sprite and setting how fast and in which direction it moves.

There are a number of 'housekeeping RSXs' which let you keep an eye on whether collisions have occurred, or edges have been met. A couple of RSXs create sounds and play these when a certain event happens on the screen. Also, there are a few commands to define the action of the computer controlled sprites.

With the Designer, there is the ability to create multi-coloured sprites up to 32 by 32 pixels; to make up sequences of these, and then to run them in sequence to check that the animation looks correct. Again, this seems to cover everything required, with facilities for mirroring, inverting and copying the sprites.

However, I do wish that extended facilities like this Designer could be called from the normal operating medium (ie from BASIC when writing the actual program) without the loss of everything else you are working on. A further limitation is that a 6128 is needed for its operation - this requirement also applies to the Compiler (see below) although the things that they produce can be run on any CPC set-up.

To make games run faster than BASIC allows you, a Compiler is available as an optional extra, which converts BASIC commands into much quicker machine code. A program can be made to run up to 16 times faster once it has been compiled, but as the manual points out, the RSXs have already been honed for maximum speed, so any savings will come from eliminating a lot of BASIC commands; the fewer there are of these, the less the apparent increase in speed.

However, the compiler leaves a lot to be desired. A great deal of rewriting of a BASIC program needs to be done before the Compiler can convert it: a few BASIC commands cannot be used and still others have to be altered. In common with many Compilers, floating point numbers (values with a decimal point) can't be handled and so integers are the order of the day; variable names are also limited & this just serves to frustrate the user.

I have never written a Sprite game before, and must confess I was disappointed with the complexity which Sprites Alive presents. However, it is a comprehensive utility, and maybe others will find its ability to handle up to 64 sprites, including missiles, just the thing they have been looking for - personally, it was just too unfriendly and complicated to really get to grips with.

SIM from Goldmark Systems (Price: £19.95 on disc)

'Sim - The Z80 Simulator', to give it its full name, is a monitor utility that allows machine code to be viewed, edited, run or single-stepped. It is intended primarily to let the user go through their, or someone else's code, and look at how it works and, indeed, whether it works. Basically a monitor program lets you simulate running a piece of code without actually carrying it out. As such, Sim sounds like an excellent debugging tool, but does it succeed...

At any location the code present is disassembled into a machine code instruction, and each of the Z80 registers contents are shown: there is even a facility for viewing the alternate register set.

One breakpoint can be set for running the code (when a breakpoint is met, the program will stop) and the area of memory, in which the code is allowed to run, can be limited. ROMs can be accessed, but not Bank RAM (see below).

Its greatest drawback is that each of its are functions separate. For example, when editing memory, to view the locations about to be altered the disassembler must first be used, stopping the display with the ESC key when enough has been shown: the new bytes are typed in, but these new values are not displayed during this process nor after it has been completed: to do this, the disassembler must again be invoked.

There also some limitations on what machine code instructions it will perform: Output ones are strictly not allowed and any alteration of RAM Banks is also not permitted. However, a useful method of running a routine in ROM is provided (to get round the problem of it being impossible to insert the break code into such memory) via the Push command.

Despite these problems, the program actually works very well - the things you can do with it are very comprehensive. The screen display produces a constantly updated list of the register contents, giving the opportunity to re-examine and alter them. Output can be directed to the printer if desired to produce a hard-copy and there are commands to load, save and merge blocks of data. A very useful command for the programmer is the 'FLAGS' instruction which decodes the flag states and presents them in a helpful manner. There are also the standard features to examine and edit memory, and switch in/out the upper and lower ROMs.

The 8-page manual is clear and helpful, although it does expect you to have a fair knowledge of machine code before you can use it properly. Overall, Sim is a useful tool for machine code programmers to test the results of their labours & is well worth the money.

GLENCO SOFTWARE - 15 Alford Lane, Whitehouse Farm, Stockton on Tees, Cleveland
SD MICROSYSTEMS - PO Box 24, Holbeach, Lincs PE12 7JF
GOLDMARK SYSTEMS - 51 Comet Road, Hatfield, Hertfordshire AL10 0SY

POKING AROUND

Lift the lid on your CPC

by Bob Taylor

So far in this series, we have been highlighting the many ways in which you can use Machine Code routines from BASIC to achieve things that would otherwise be extremely difficult. There has been a drawback - some routines need to have bits of data to work on, and in specific places; this is impossible to achieve straight from BASIC and therefore, we have printed in this issue a BASIC program which, when run, will allow you to assign data/information to any of the registers used by Machine Code.

PASSING PARAMETERS

When we use a CALL command or an RSX from BASIC we can pass up to 32 pieces of information to the Machine Code routine by entering such data as parameters that follow the RSX or CALL instruction.On entry to such a routine the parameters are held in a block of data. This block is arranged with the IX register pointing to the last parameter present; earlier parameters are placed in sequence above this last one, up to the first one we entered.

On entry to these routines, the A register holds the number of parameters that have been entered. Each parameter is represented by two bytes which could be:

a) **THE ADDRESS OF A STRING DESCRIPTOR.** At this address, will be three bytes of data to represent a string expression (a string variable only, in the case of the 464). These three bytes are as follows:

BYTE 1: the length of the string

BYTES 2 and 3: the address of the start of the string - note that this address is not the same as that of the string descriptor which points to the length byte above)

b) **THE ADDRESS OF THE VALUE OF A NUMERIC VARIABLE.** This variable could be Integer or Real, so the value will be present there in two or five byte form.

c) **THE VALUE OF A NUMERIC EXPRESSION.** This will be in 2 byte form, even though any part of the expression could be in Real/Floating Point (5 byte) form.

THE FIRMWARE

When we consider 'Firmware Calls', the situation becomes even more difficult. With these, any parameters required (called entry conditions) need to be present in specific Microprocessor registers and not as a block of data to be picked out when needed.

Some Calls need no entry conditions, and so can be called with no problem (for example CALL &BD19), but usually this is not the case.

There are some calls which only need a value in the A register; providing this value is less than 33, we can take advantage of the fact mentioned earlier that on entering a routine from BASIC, the A register holds the number of parameters present. Try CALL &BC0E,0,0 to change the screen Mode to 2, then try CALL &BC0E to change it to Mode 0. In the first case, there were two parameters present, so A held &02 on entry to the routine & this made the routine change to Mode 2. The second CALL had no parameters so A was &00, resulting in Mode 0.

With higher values required in A (up to 32), the command looks pretty unwieldy with all those parameters, but you can still use this method for some Calls. By the way anything can be used here for a parameter - ie a string (6128 only), a string or numeric variable, or a number.

But what of the others. Unfortunately, the remaining registers are set by the Operating System for its needs, and so there is very little scope allowed us to utilise ROM routines in this way.

ROMCALL & RAMCALL

The RSX presented here allows each register to be loaded with required values on entering a machine code routine. It has two versions:

|RAMCALL is for using Firmware Calls (whose entry points are all in RAM).

|ROMCALL is for accessing any useful routines in the various ROMs.

Values to go into registers are entered as optional parameters. Both RSXs have a similar sequence of parameters.

The only difference is that the first parameter for |ROMCALL has to be the ROM select number of the ROM which contains the required routine; to access routines the Lower ROM, use a value of -1 here. This is followed by the actual address of the routine in ROM.

With |RAMCALL, this address (of the 'Firmware entry point' in RAM) will be the first parameter.

After these obligatory one or two parameters, come the values we wish to place in the various registers. The sequence of registers that I decided on - see the section on syntax below - is not the standard alphabetical one but instead it is based on frequency of use.

On entry to an Upper ROM routine, the IY register is given the address of that ROM's reserved work area by the Operating System, so any IY values entered with |ROMCALL may be lost; I have included the option in case the Lower ROM, or a RAM routine of your own, needs it.

Register parameters are optional with the proviso that if a certain register's parameter is required, then all those before it must also be entered. Any value will do for these unwanted earlier registers; any later unentered registers will be filled with &0000. The first (A) register parameter also allows values to be entered to simulate Flags (Carry, Zero etc) - if required, the relevant Flag bit value should be multiplied by 256 and this added to the A register value.

Note that values for B, D, and H registers will also need to be multiplied by 256, but not those for C, E, or L.

GIVING THE DATA

The Syntax for each RSX is as follows:

```
|ROMCALL ,ROM Select No ,Address of routine [[[[[[,FA (note the reversal of
          the usual AF)] ,HL] ,DE] ,BC] ,IY] ,IX]
|RAMCALL ,Address of Firmware Call [[[[[[,FA] ,HL] ,DE] ,BC] ,IY] ,IX]
```

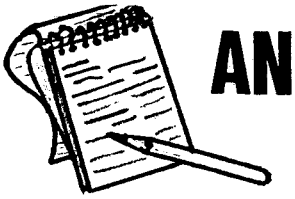
Passing register values out of the Called routine could have been accommodated by use of the '@' operator with numeric variables, but this would have increased the number of parameters necessary, or have forced the use of variables instead of numeric expressions. To keep the parameter situation simple, I decided to use an output data block at &BEF0 which maintains the input parameter sequence:

&BEF0	&BEF1	&BEF2	&BEF3	&BEF4	&BEF5	&BEF6	&BEF7	&BEF8/9	&BEFA/B
F	A	L	H	E	D	C	B	IY	IX

These locations may or may not hold valid data depending on the exit conditions of the Called routine. The loader routine for the RSXs is given below.

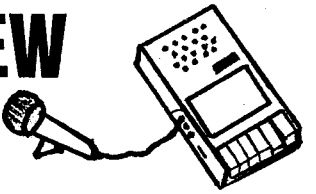
PROGRAM:

```
[F1] 10 'ROM/RAM-CALL Loader  by Bob Taylor  (copyright 1991)
[20] 20 MEMORY &7FFF:RESTORE:PRINT:PRINT"Please wait a few seconds"
[BA] 30 FOR lin=0 TO &B0/8-1: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$)<>total THEN PRINT:PRINT"Error in line"lin*10+110
        :PRINT:END
[94] 70 NEXT lin:IF PEEK(6)=&80 THEN POKE &80A4,&55:POKE &80A5,&CB
[BF] 80 PRINT:PRINT"All M/C loaded":PRINT:PRINT"Press S to save M/C as
        ROMCALL.BIN":PRINT"or any other key to continue":WHILE INKEY$="":WEND:
        IF INKEY(60)<>-1 THEN SAVE "ROMCALL.BIN",B,&8000,&A9
[65] 90 PRINT:PRINT"To Load and Initialise |ROMCALL RSX just use:":PRINT"MEMORY
        HIMEM-&A9:a=HIMEM+1:LOAD"CHR$(34)"ROMCALL.BIN"CHR$(34)",a:CALL a":PRINT
        "with the Disc or Tape inserted"
[EA] 100 END
[CA] 110 DATA D5,62,6B,36,C9,01,18,00,2BA
[07] 120 DATA 09,EB,01,28,00,09,72,2B,1C3
[4B] 130 DATA 73,44,4D,E1,23,C3,D1,BC,458
[E6] 140 DATA 52,4F,4D,43,41,4C,CC,52,2DC
[D8] 150 DATA 41,4D,43,41,4C,CC,00,18,242
[C8] 160 DATA 80,3D,F6,80,32,FF,BE,E6,508
[1A] 170 DATA 3F,28,6C,47,2F,C6,08,28,23F
[82] 180 DATA 09,30,64,21,00,00,E5,3D,1E0
[F8] 190 DATA 20,FC,DD,66,01,DD,6E,00,3AB
[8D] 200 DATA E5,DD,23,DD,23,10,F3,E1,4C9
[48] 210 DATA 22,FD,BE,E1,7C,65,6F,E5,4F3
[FA] 220 DATA F1,E1,D1,C1,FD,E1,F5,3A,671
[FE] 230 DATA FF,BE,CB,7F,28,13,DD,7E,49D
[37] 240 DATA 00,32,FF,BE,3C,3E,CF,28,360
[14] 250 DATA 0A,F1,DD,E1,DF,FD,BE,18,56B
[1A] 260 DATA 0B,3E,C3,32,FC,BE,F1,DD,4C6
[C8] 270 DATA E1,CD,FC,BE,22,F2,BE,F5,62F
[3D] 280 DATA E1,65,6F,22,F0,BE,ED,53,4C5
[76] 290 DATA F4,BE,ED,43,F6,BE,FD,22,5B5
[6B] 300 DATA F8,BE,DD,22,FA,BE,C9,3E,574
[60] 310 DATA 21,0E,00,21,93,CA,C3,1B,28B
[75] 320 DATA 00,00,00,00,00,00,00,00,000
```



AN INDUSTRY INTERVIEW

with STEVE DENSON



As from this issue, we are going to be running a series of interviews with some of the people connected with CPC and Plus computers. To start off with, we asked Steve Denson, the owner of SD Microsystems, for his thoughts on the CPC market, and here are his answers.

Do you feel that Amstrad's decision to produce a new CPC-compatible computer was a wise idea, or would they have been better advised to concentrate on new 16-bit computers?

The CPC computer has been immensely popular & so I feel that Amstrad were quite right to launch a successor. Of course the ideal machine would have been both 16-bit and CPC compatible, building on a huge existing software base but I suppose that technically this was impossible.Yet I still think that the Plus range was a case of good product and bad pricing.The top of the range 6128 Plus should have gone into the shops at around £299 & then with its colour monitor and sleek looks it might well have been a winner during last Christmas. The marketing strategy was also wrong. The CPC has always been a strong all-rounder and not just a 'games machine', which is where the Plus was positioned. Interestingly, Atari are now pitching the ST as a 'general-purpose' computer.

It has been said that the Plus computers and the swift demise of the GX4000 have harmed the CPC range & encouraged people to call all 8-bit computers 'obsolete'. Would you agree with this?

Yes. I think the relative failure of the Plus range has harmed the 8-bit market which is a shame because if a computer does a good job, what does it matter whether it is a 16-bit or not? There is tremendous snobbery in computing and some people are obsessed with mega-memory and super-speed. Anything less that the latest gadget won't do for them and unfortunately this type of attitude is often encouraged by the computer press.

For how long can you see the CPC computer market surviving and do you still feel that there is a demand for 'serious' products? Do you find the leading newsstand magazines helpful in promoting the serious side of CPC computing?

The demand for CPC related products will last as long as people are using the machines in large enough numbers and I think the market will continue for some time to come. Even if Amstrad cease production, the second-hand sales will continue to thrive, and so there are always new and potential customers. No, I don't think the professional Amstrad magazines do enough to promote serious software and I've told them so often enough! AA especially seems to be getting very gamesy & I think the worst move they ever made was to remove the Buyer's Guide.

Why did you decide to write software for the CPC computers?

I started writing application software for the CPC simply because it was, and is, such a capable computer. We have everything from home accounts to company payroll software in our range and I think we've proved that small businesses and serious home users don't need to invest in an expensive PC system.In fact the whole concept of SD Microsystems is to offer a cheaper and simpler alternative to mainstream computing and we have achieved this with the Sinclair QL, Amstrad PCW and, of course, the CPC which is still our No. 1 machine.

Have you any new plans for software or hardware in the pipeline?

We certainly do have plans for the CPC - these include a simply stunning art package that will set new standards in graphics & a 3.5" second drive hardware/software bundle that we hope to have on the market by Christmas.

For how long can you foresee firms, such as yourself, being able to support the CPC market?

We will continue to sell and to support our products as long as there is demand for them which I expect to be for some years yet. PD software is a bit of a fly in the ointment for companies like ourselves who are trying to sell full-price serious software but I feel we can beat most of it on quality & professional support.

What direction do you see the market taking in the next few years?

To be honest, I think computing will become rather more predictable than exciting. Why is this? Because the industry is becoming more and more standardised. Even Apple are getting together with IBM to make compatible computers in the future. The days of the old format-wars, when each rival manufacturer developed their own operating system are long gone. Maybe it wasn't a very practical approach but it sure was fun!

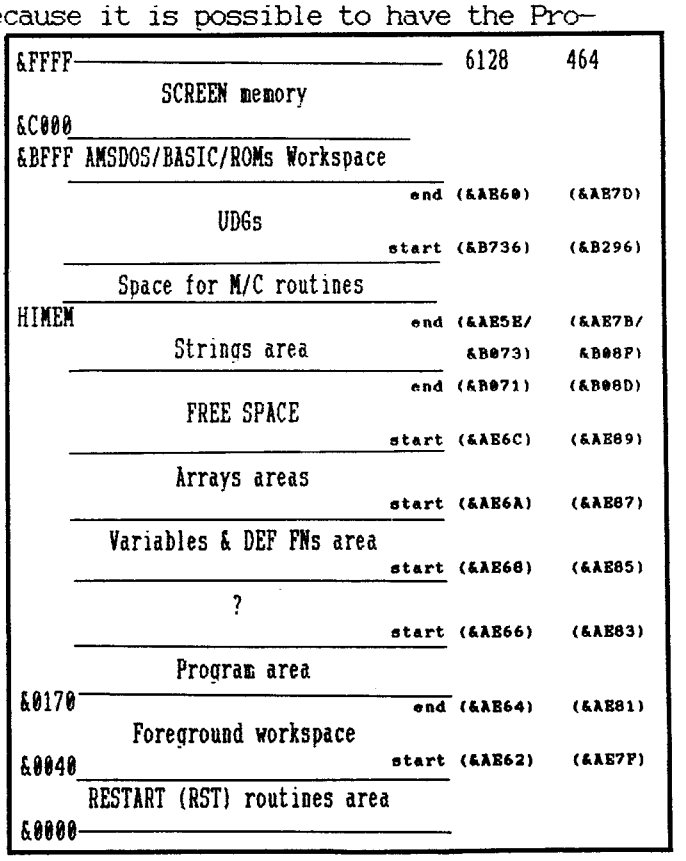
BASIC Tokens (4)

Having looked, in previous issues, at the way that the Operating System uses areas of memory for storing a Program, Variables and Arrays, this month we will complete the picture with coverage of the Strings area.

MEMORY MAP

Before we delve into the details of the Strings area, it's best to try to get an overall picture of the order and place in memory of all the areas. At any one time there is only 64k of RAM memory accessible. The top 16K is usually used for the Screen, and of the remaining 48K, areas at the top and bottom are reserved for the Operating System to carry out its housekeeping. What's left is available for BASIC Programs, Arrays and so on. It's possible for sideways ROMs to reserve for themselves some of the top and bottom areas of this remainder but since this would complicate things even more we shall exclude this at present.

The first available memory location is at &0170 and it is at this address that the Program should start. I say 'should' because it is possible to have the Program area higher up in memory. In the top reserved area, there are Addresses stored which tell the Operating System where such things as the Program and Variables areas start - these Addresses (which are called System Variables) can be altered by poking & so our Program could be situated higher in memory. This is something which you may never come across but it's possible if you want to have several programs resident in memory at the same time: the organisation needed to avoid upper programs being overwritten by lower ones is quite complicated but it could be done.



Anyway let us assume that our Program is normal and starts at &0170. Of course, the more Program we type in, the more room it takes up. After the End of Line character for the last line, there are 2 empty bytes that are meant to be like a following line of zero length.As we saw in the article on Tokens this indicates that there's no more program left (unless we have deceived the computer) and is used by some program scanning routines to detect the end of Program - the end can also be found from one of the Addresses mentioned above.

Immediately following these two empty bytes is the start of the Variables area but in between there is the .... area. There may or may not be something that is meant to go here but if there is I have never seen it. How do I know it's there?

Well, the Addresses mentioned above are arranged in sequence and between that of the start of the Program and that of the start of the Variables is another which always holds a copy of the Variables start: either it's the address of the start of another area (which never seems to hold anything) or it's superfluous & could have been left out - I rather think it is the latter.

So then, normally the Variables area follows the Program area directly and in turn is followed by the Arrays area. If we add to the Program then any Variables and Arrays are moved up, as an intact block, in order to make room. Likewise the Arrays area is moved up if we add a Variable to the Variable area. And similarly if we reduce the length of the Program then both the Variables and Arrays areas are moved down in memory.

Next comes an undesignated area, which ends at the start of the Strings Area. This area gets smaller as the Program, Variables, Arrays or Strings areas become larger, and expands again if any if them should be reduced - for example, when a line is DELETEd or CLEAR is used. By the way, both CLEAR and RUN reduce the Variables, Arrays and Strings areas to nothing - they work by resetting the System Variable Addresses to their minimums with respect to their neighbours and not by erasing the contents of the areas which are left largely intact. NEW is entirely different and wipes clean the whole of RAM from the Program start to the end of Strings, setting each byte to &00 - there is no going back from a NEW, unlike on some other computers.

The undesignated area is not unused however. For example the top 2K is used by the Disc Operating System AMSDOS when LOADing from or SAVEing to disc. Sometimes this free area is reduced to less than the 2K required, due to the length of the Program or the number of Variables or Arrays, and under these circumstances any AMSDOS operations will result in a 'Memory Full' Error message.

The Strings area is the last one to be considered and extends from the address we have set with a MEMORY command (or the default one which is set on switching on the computer) downwards to the end of the free area which is where the String Area begins.Downwards is the operative word due to the fact that new strings are added to this area at its lower edge & as it acquires more strings so it extends downwards.

Just to complete the picture, but not as part of our deliberations, above the Strings area comes any area we might have set aside for Machine Code routines by invoking the MEMORY command, and above this are any User Designed Graphics that we may have (at Switch On there are 32 character matrices here occupying 8 bytes each and this number of characters may be reduced to zero or increased to 224 by using the SYMBOL AFTER command). Either or both of these areas could be reduced to nothing. Above this, comes the area that is reserved by the Operating System, which is another story altogether.

STRINGS AREA

The Strings area is used for storing only some of the strings that are used in BASIC.Any complete string to be printed doesn't need to be stored here & neither does a delimited string assigned to become the whole of a String Variable or an Array element. Strings which are stored here are:

- those used as parameters for RSX's or CALL commands
- those whose String Descriptor addresses have been used by invoking the '@' operator (not on a 464)
- strings that are formed by concatenation (ie. by joining together) whether they are to be assigned to a variable or just PRINTed
- any string that is a part of another string obtained by using the Functions LEFT$, MID$ or RIGHT$, even if such a slice is ultimately to become part of a concatenated string; this kind of slice exists as a separate entry in the Strings area as well as forming part of another string entry (not 464)

In a program, on encountering a string included in the list above, the Parser stores the string itself below any last one entered. So the Strings area expands downwards. The string itself is stored the correct way round and not in reverse. No attempt is made to put the strings into any sequence which is related to the name of the relevant Variable or Array - it is strictly on a first-come, first-served basis (as are the Variables and Arrays areas) - and so strings belonging to an array will not be in the same order as the elements to which they apply, except by (reverse) coincidence.

If the contents of a String variable or Array element are altered then any old entry in the Strings area related to it will be abandoned and if its new assignment warrants it a new entry will be made here for it. If much use of strings is made in a Program then the Strings area could become full of unusable discarded strings and this could eventually lead to the computer halting the program it is running for a period while it retrieves the space occupied by such entries. If you are running such a program it is advisable to regularly use a statement that contains 'FRE("")' in order to force a 'garbage collection' which dispenses with such useless strings leaving only the currently used versions. This should prevent the halting of the program mentioned earlier.

The sequence for an entry in this area differs between the 464 and the 6128/664:

TWO LENGTH BYTES (on 6128/664 only). A string can never be longer than 255 characters so the second byte provided here will always contain 0. The first byte should match the length given in the string descriptor of the Variable or the Array element. A test on the contents of these 2 bytes (in the Variable/Array areas & in the Strings area) is useful for checking whether a parameter which is passed to an RSX is of the string type - it's not infallible but if there is a discrepancy then the parameter cannot be a string (of course, this test cannot be applied to the 464).

THE STRING ITSELF containing every character exactly as assigned.The location of the first byte of this string is the address given in the string descriptor.

You can use the relevant version of the listing below, with your own lines, to show examples of the use of the Strings area.
For a 6128 only, the entry's print-out will start with the address of the length byte and be followed by all the bytes for that entry from the length byte to the last byte of the string proper. Because an entry does not need to have a string Variable associated with it, no Names are given - they wouldn't be in this area anyway. Note the use of '@' in line 1 to force 'a$' into the Strings area. Only four of the eight strings assigned are present.
With the 464, notice that even fewer of the eight strings can actually be found in this area and that these two strings run into each other.

```
[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)
[C6] 70 DIM q$(2):q$(0)="1st":q$(1)="2nd":q$(2)="3rd"
[C1] 1300 DEFREAL n:n$="N"
[7F] 1310 FOR n=PEEK(@n$+1)+256*PEEK(@n$+2)-2 TO HIMEM
[80] 1320 PRINT:PRINT"&"HEX$(n,4)" ";
[37] 1360 FOR n=n TO n+PEEK(n)+1
[E3] 1370 PRINT " "HEX$(PEEK(n),2);
[8F] 1380 NEXT:n=n-1
[0D] 1390 NEXT
```

For the 464, omit line 1380 and alter the following lines:

```
[F6] 1310 n=PEEK(&B08D)+256*PEEK(&B08E) '&B08D/E is the system variable holding
          the address of the end of the Free Space area
[EB] 1360 FOR n=n+1 TO HIMEM
```

This listing can be MERGEd (after being SAVEd) with listings 2 and 3 from the previous article on Variables and Arrays, and will then show which String Descriptor addresses point to strings in the Strings area and which to strings in the program area. Listing 1 from the article on Tokens can also be merged to provide a more complete picture of the complicated shenanigans involved when running any BASIC program - now you can start to see why BASIC runs comparatively slowly.

Mail Order Problems

Many thanks to everyone who wrote in response to the section on Mail Order in last issue's editorial. It seems that the problem is much more widespread than I had imagined. For legal reasons we are prevented from releasing the names of any companies we have heard of, until all cases against these firms have been looked into. However if you do have trouble with a company we advise you to contact the Office of Fair Trading, the Advertising Standards Authority and, if you have no joy, ask your solicitor to write a letter to the company concerned, threating to take sue them through the 'Small Claims Court'. If you want any more information on this, then get in touch and I will pass on further details of what procedure to follow in order to get your money back.

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!

PLAYMATES - Issue 9 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.

DISC SUITE - A comprehensive disc utility, packed with loads of useful features, including a special ASCII helper plus free PD, if you mention Print-Out. Send £4.99 and disc to Adrian Sill, 19 Sherwood Drive, Skellow, Doncaster, South Yorkshire DN6 8NY.

FOR SALE - CPM on 2 ROMs + instructions (£25), WOPS Disc Utility (£5), AMX Stop Press including mouse (£50), SuperCalc Two under CPM (£25), Rembrant Paint Disc (£7), Extra Extra for Stop Press (£10), Tasword 6128 and Taspell Disc (£22 for both).2-IN-1 CPC-PC-CPC file transfer (CPM) by Moonstone (£20).
DISC GAMES (£6 each): Yes Prime Minister, Trivial Pursuit, Pegasus, PHM, Mini Office 2, Driller, Dark Side, Total Eclipse, Hunt for Red October, Captain Blood.
TAPE GAMES (£3.50 each): 5 Star Games One, 5 Star Games Three, Night Raider, Spitting Image, TV Special. TAPE GAMES (£0.50 each): Nexor, Red Arrows, Dambusters, Jump Jet, Fighter Pilot.
For any of the above contact: Phill Mackay, 12 Lydstep Road, Barry, South Glamorgan CF6 3EB, or 'phone (0446) 721289

FOR SALE - Operating Amstrad CPM 2.2 (£5), and also back issues of AA, CwtA and ACU. Send an SAE for a list to: M. Pinder, 4 Wham Hey, New Longton, Preston PR4 4XU.

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 features:

24-PIN PRINTER DUMPS
- Following on from our highly popular printing routine in Issue Nine, we now have a shaded screen dump version for 24-pin printers.

PD LIBRARIES
- We begin our round-up of the libraries and make our recommendations for the software you've just got to own

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 program, 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.

CHRIS – Help given on anything: No problem too small; also PD documents printed out, just send disc with DOC files on & I'll send DOCs back printed out (3p per page both sides) with your disc and a bill for postage. Okay! Write to: 6 Frank Street, Great Horton, Bradford BD7 3BT.

SEAN McMANUS – Help given on all aspects of BASIC & assembly language programming but don't forget the SAE!!! Get in touch at: 226 Chertsey Rise, Stevenage, Herts SG2 93Q.

ALAN SCULLY – Help offered on BASIC programming (fairly advanced), printers, all aspects of Public Domain, disc to disc copying and virtually everything else from hardware advice to finding a piece of software; I'm useless at CPM though! You can get in touch with me at: 119 Laurel Drive, East Kilbride, Glasgow G75 9JG.

SAM WRIGHT – I want to get in touch with anyone with ROM blowing or file copying experience. Write to: 24 Chester Avenue, Whitehead, Carrickfergus, Co Antrim BT38 9QQ.

If you want to be included in this helpline, just complete the form and send it off to us at the usual address. 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. ☐