THE AMSTRAD USER

Issue No. 14 $3.50 March 1986

- *MUSIC COMPETITION - WIN a Radio/Cassette Player*
- *BANK SWITCHING DEMONSTRATION on the 6128*
- *A NEW SERIES ON CP/M FOR BEGINNERS*
- *USER GROUP INFORMATION*

FOR THE NOVICE & EXPERIENCED USER

Registered by Australia Post – Publication No VBP7017

THE AMSTRAD USER

Issue No. 14
March 1986

Editorial ____ 2
Letters ____ 3
CP/M Explored - Part 1
by Shane Kelly ____ 5
6128 Segment
by Mark Godden ____ 9
Towers of Logo
by Jeff Hughes ____ 11
User Group Information ____ 13
Hints for high score games ____ 15
"March Madness" Book Sale ____ 16
Software Reviews ____ 17
The Learning Centre
Introduction to Music - Final Part
by Peter Campbell ____ 19
Music Competition
Win a Radio/Cassette player ____ 24
Book Reviews ____ 25
Disc Cataloguer
by Petr Lukes ____ 27
Disc Drive and Binder Offers ____ 31

For Tape Subscribers, the programs can be found at these approximate positions:
Side 1: BANKCAT- 6, HANOI1- 27, HANOI4- 36, MLIST45- 56, MLIST67- 95
Side 2: DISKCAT- 6

All enquiries and contacts concerning this Publication should be made to The Amstrad User, Suite 1, 33 The Centreway, Blackburn Road, Mt. Waverley, Victoria 3149, Australia. [Telephone: (03) 232 7055].

The Amstrad User is published each month by Strategy Publications.

The single copy price of $3.50 is the recommended retail price only.

The subscription rate (for Australia only) is $35.00 for 12 issues of the magazine only, or $75.00 for 12 issues of the magazine plus tape containing all programs appearing in that issue. Postage is included in the above prices. Overseas prices available upon application.

Please note that whilst every effort is made to ensure the accuracy of all features and listings herein, we cannot accept any liability whatsoever for any mistakes or misprints.

Contributions are welcome from readers or other interested parties.

In most circumstances the following payments will apply to published material: Letters-$5.00, Cartoons-$5.00 and a rate of $10.00 per page for programs, articles etc.

Contributions will not be returned unless specifically requested coupled with suitable stamped and addressed padded bag (for tapes) or envelope.

THE AMSTRAD USER

G'day,

Our first Birthday passed unceremoniously last month yet we were inundated with cards. No, not Birthday cards - renewal cards for subscriptions that had originally started with our first edition in February 1985. It makes my job all the more satisfying to know that the magazine has been accepted so well by so many Amstrad Users.

The CPC6128 has, by all accounts, achieved amazing sales since its launch a few months ago, and naturally enough, it will take a little while before contributions start to filter through which are specific to this machine. However, to satisfy those owners now, we feature a new column called '6128 Segment'. This will hopefully be the start of a larger coverage and, of course, will depend upon the contributions received.

We also feature for the first time a program in LOGO. As everyone who owns an Amstrad disc drive has a copy of LOGO, I am surprised it has taken so long for it to get a mention. Many Primary schools use LOGO as an introduction to computers. Perhaps you have developed some procedures which may interest our younger readers or stupefy our older ones!

Our Disc Drive offer continues, providing a saving of around $100 on the cost of a DDI-1 and many subscribers have jumped at the opportunity. This month's special (for March only) is a go!den opportunity to get that book you wanted at a price which is getting closer to cost than ever before. As usual, we don't have limitless supplies, but we will try to satisfy everyone's needs.

Finally, while I am in a 'give-away' mood, you may wish to enter our Music Competition on Page 24. If you don't feel qualified enough then I suggest you read the last six Learning Centre articles on Music. Peter Campbell's work on these articles certainly taught me a lot, and I am grateful for his effort in compiling this long series. Give it a shot - as the blurb says "it needn't be complicated".

See you next month,

Ed

Letters

All correspondence published in this section earns a payment of five dollars.
Letters should be addressed to The Editor, The Amstrad User, Suite 1, 33 The Centreway, Mt. Waverley, Victoria, 3149.

After many months of research into what computer I should buy my children for Christmas, I chose the Amstrad 6128. There were many factors that influenced my decision, one being the service described by letters to the Editor in your magazine by AWA-Thorn, however, I never thought I would need service in the short term. How wrong I was.

On Christmas Eve when I was carefully putting computer, monitor and printer together I was not able to operate the disc drive successfully. Everything I asked it to do came back with 'bad command'. Hoping for some kind of miracle before the kids woke up Christmas morning, I tried it every hour. How disappointed they were when they couldn't use the software I had purchased.

Thinking still of a miracle, I phoned my local Amstrad dealer at 9 a.m. where I purchased my computer and asked him for advice. When he was unable to help me on the phone he said he would see me shortly and within half an hour was on my doorstep. Unfortunately he was not able to fix the disc drive, but I did appreciate his concern. The dealer apologised for nor being able to replace it immediately owing to being sold out of stock but advised me to return it the first working day after Christmas and AWA-Thorn would replace it with a new machine. When I arrived with my faulty computer, my dealer had a new one ready to go.

Although Christmas day was somewhat of a disappointment, I certainly appreciate the excellent service I received from my Amstrad dealer.

Brian Miller, Hope Valley, SA

Our hats off to North-East Computers of Redwood Park for providing a happy ending to the story - Ed.

During the Christmas period I received a new Amstrad 6128 after trading in my CPC464. Both computers are exceptional, though the 6128 is far superior.

Most of my software was for the 464 and on tape. Do you know if anyone or any companies convert tapes to disc?

Anthony Greenland, Uralla, NSW.

The short answer is 'yes' - but they are doing it illegally. When you bought your software for the 464 you were buying a licence to use it in that medium only. Unless authorised, converting the software from tape to disc is breaking the copyright protection which, if you read last month's article on the subject, could land you in jail for up to two years and/or a maximum fine of $50,000. It's probably cheaper in the long run to purchase a standard cassette player and cassette interface lead (CL-1) to run your current software and where possible buy discs in future.

I read with some interest the article "Menu Utility" by Reuben Carlsen in TAU Issue No. 10. I agree with the alpha-key selection idea as do many commercial software writers. However, I cannot see the sense in using A, B, C, D and so on when the letters do not apply meaningfully to the menu item. One may as well go back to 1, 2, 3

etc. Reuben and your other readers may be interested in the following approach which, I am sure, has been thought of by many others but was an 'original' idea for me.

Allow me to explain be means of an example. The following demonstration program simply allows the operator to select the screen border colour via a menu routine.

```
10    MODE 1:INK 0,0
20    CLS:PRINT"Which Border
      Colour ..."
30    PRINT
40    PRINT"Red . . . . . R"
50    PRINT"Orange . . .  O"
60    PRINT"Yellow . . . .Y"
70    PRINT"Green . . . . G"
80    PRINT"Blue . . . . .B"
90    PRINT"Indigo . . . .I"
100   PRINT"Violet . . . .V"
110   PRINT
120   PRINT"Press a Letter"
130   k$=LOWER$(INKEY$):
      IF k$="" THEN 130
140   x=INSTR(roygbiv"",k$)
150   ON x GOTO1000,2000,3000,
      4000,5000,6000,7000
160   GOTO 130
1000  BORDER 3:GOTO 20
2000  BORDER 15:GOTO 20
3000  BORDER 12:GOTO 20
4000  BORDER 9:GOTO 20
5000  BORDER 11:GOTO 20
6000  BORDER 1:GOTO 20
7000  BORDER 7:GOTO 20
```

Lines 10 to 120: simply set up the screen and print the menu options.
Line 130: scans the keyboard and returns, in k$, the LOWER case equivalent of whatever key was pressed.
Line 140: tests k$ to see if it is a sub-string of "roygbiv", the initials of the colours offered by the menu in the same order. If, for example, the operator pressed "y" (or "Y") then the INSTR function will give x the value of 3 since "y" is contained in "roygbiv" at the third character. If some key other than the 7 operative ones is pressed then x will be given the value of 0. INSTR is a very powerful function for this sort of processing.
Line 150: uses the ON GOTO command to access the appropriate part of the program. If "y" was pressed then control passes to line 3000. ON GOSUB would work equally well with a RETURN in each subroutine.
Line 160: is a "trap" for non-operative keys being pressed either deliberately or in error. When x has a value of 0 the ON GOTO (or ON GOSUB) commands are ignored and control passes to the next command, in this case a GOTO 130.
Lines 1000 to 7000: are the routines to carry out the menu statements and should be self explanatory.

Of course, the above is only a short demonstration to illustrate the idea and in any other program the letters would depend upon the menu choices. Occasionally, two choices will have the same initial and here you will need to think of an alternative name for one of them. For example "SAVE TO DISC" and "SORT" present a problem. A possible solution would be to use "ORDER" instead of "SORT".

In theory, it would be possible to have up to 26 choices (or even more if you are willing to use numbers and special characters!) each selected by pressing a single key.

Roy Lundquest, N. R'hampton, QLD.

I have discovered some faults in the program "Space Explorer" in The Amstrad User for January 1986. These are the transposition of the horizontal and vertical coordinates of the map; some of the planet positions not in agreement with the textual descriptions and the first part of the program which displays the encyclopedia of the planets not being connected to the quiz of the space explorer.

The corrections that I have found desirable are set out in the new lines below:

```
765   DIM Q(10): Q=0
780   BORDER 20: Q=Q+1: IF Q=10
      GOTO 4050
1020  MOVE 490, 90
1030  PRINT CHR$(230);
1040  MOVE 440, 190
1060  MOVE 240, 240
1140  MOVE 340, 290
1160  MOVE 300, 300: PRINT
      CHR$(231);
1660  INPUT "Plot horizontal
      coordinates please (1-
      9)";V$
1680  INPUT "Plot vertical
      coordinates please (1-
      8)";H
2180  IF H=5 AND V$="E" OR
      V$="e" THEN GOSUB 3000
2190  IF H=6 AND V$="G" OR
      V$="g" THEN GOSUB 3310
```

H.L. Bailey, Coffs Harbour, NSW.

I have just finished reading the November 1985 issue of The Amstrad User and have found it most informative. However there is one question I would like answered before I commit my $$'s to a subscription.

You have two types of subscription, one just the magazine ($35) and another the magazine plus cassette ($75). I was wondering if you are going to make a similar offer now that there are so many 664's and 6128's around plus 464's with additional disc drives.

L. Knowles, Wallangarra, QLD

To supply a disc with a magazine each month could be done, but the final cost would be prohibitive. Multiplying the cost of one disc by 12 and adding it to the standard $35 subscription plus the cost of mailing will give you some idea of our reluctance!

However, we are currently looking at what we can do for disc-only users, and are toying with the idea of producing a disc each quarter containing the previous three months published software. We may also produce a disc with the entire first year's programs on it. It is estimated that a disc subscription will cost $95 per year (4 discs and 12 magazines) and the "year disc" $50. To all readers, let me know what you think before we make the final decision - Ed.

CP/M Explored - Part 1

By Shane Kelly

So you now have a disc drive for your Amstrad? (464 or 664 - it makes no difference). You have no doubt read the DDI-1 Manual and it didn't make much sense at first. It probably got clearer on the second reading and perhaps light was dawning on the third. If so, you are now adept at using the drive as a speedy cassette, but perhaps you realise there is more to this animal than just being a replacement cassette unit. You are right!

What these articles are aiming to do is teach you to realise the convenience that disc drives can bring to your computing whether it be just for the hobbyist or for the professional user. First we have to understand what the disc drive can do in it's simplest form. This can be summed up in a simple sentence. The disc drive takes a stream of bits directed to it by the CPU and stores it on and retrieves it from a magnetic disc. That is all the physical disc drive can do. Where it stores those bits, how it organises the space on the disc, how it knows a disc is there are all functions of a program that is almost invisible to the user. It is this program that gives the disc drive it's personality and flavour. It is this program that shapes our thinking when we need to access the stored bit stream on the disc. This program is the *OPERATING SYSTEM*.

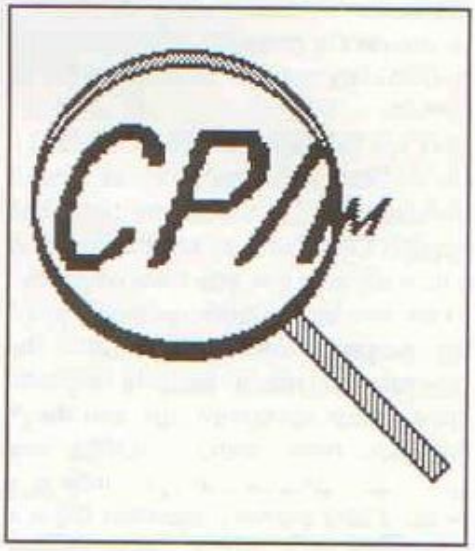
There are as many operating systems as there are stars in the sky (well almost!), and the Amstrad has two. The one you will probably be most familiar with is *AMSDOS*. You are using Amsdos when you use any external command (that is one preceded by a bar "|" that accesses the disc drive. |DIR, |ERA, |USER, |REN etc. are all Amsdos commands.

The other operating system is CP/M. Fortunately for us, Amstrad have made Amsdos a subset of CP/M so that we don't have to learn two separate operating systems, only the points where one differs from the other. The major difference between CP/M and Amsdos is that while using Amsdos you have the on-board basic interpreter as the main program in use. This means that not only is Amsdos a subset of CP/M, it is also only operative while using Basic. (For the purpose of this article any machine code subroutines run under Basic are considered to be /BASIC/ programs). Therefore if you are using CP/M, no Amsdos commands are available and vice-versa. This is not a great handicap as the the commands used under both are almost identical.

Let's take for example the command to rename a program. Under Amsdos it is :-

```
|REN,@A$,@B$
```

(Rename to A$ the program now named B$)



Under CP/M it is :-

```
REN FRED.BAS=BILL.BAS
```

(Rename to FRED.BAS the program now named BILL.BAS)

You can see that the general form is the same and may be written in general like this :-

COMMANDNAME *(parameter) (separator) (parameter)* - where the separator and parameter may be repeated as necessary. Here are some examples:

```
REN FRED.BAS=BILL.BAS
ERA FRED.BAS
DIR A:
DIR B:
USER 1
```

All these are valid CP/M commands and, with the appropriate changes in syntax, are also valid Amsdos commands.

Right, so far I have done all the work. Now it is your turn. Look at the DDI-1 manual chapter 3 and read 3.1 the introduction. OK, now turn to chapter 1 page 1.1. Follow the directions to make a back-up master disc in section 1.1. If it doesn't make sense then do the following:

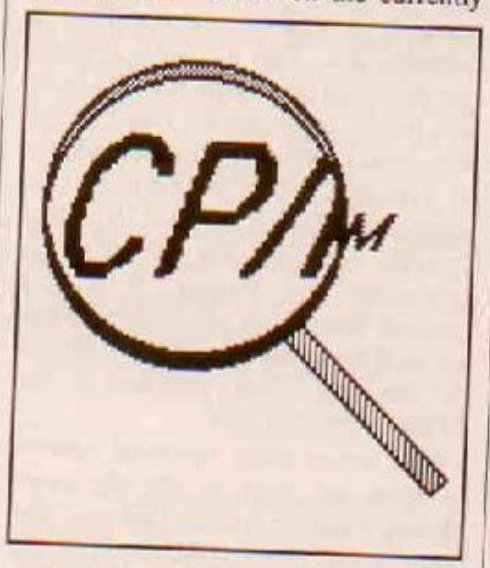
1. Reset the computer using shift CTRL escape. You are now in Amsdos
2. Insert your original CP/M disc into drive A, side A uppermost
3. Type |CPM and press return. The screen will go blue and then CP/M will announce itself in black on cyan writing. Below the sign on message will be the prompt "A>".
4. Prepare a new disc making sure it is ready to receive data (see F2 in manual)
5. Type DISCCOPY and press return
6. Follow the directions on screen remembering that the source disc is the original CP/M disc and the destination disc is the new one you have prepared.

You now have a working disc copy of the original CP/M disc. Take the original and put it back in it's safe place. Insert the copied disc into the A drive and type control C (CTRL key and C key simultaneously). There is a burst of disc activity and then the A> reappears. This is the normal screen appearance of CP/M. Whenever the A> appears, CP/M is ready to take your commands.

There are two types of commands available in CP/M. They are named *built-in commands* and *transient commands*. The difference is that the built-in commands do not have to be loaded from disc before working. All transient commands must first be loaded into memory and then executed.

The built-in commands are:-

LETTER: where letter is the drive letter corresponding to the disc drive to be selected. Note that the Amstrad allows only two disc drives in the system A: and B: while CP/M allows drives in the range A to P. As an exercise type C: and press return. You will get the cryptic message BDOS ERR ON C: SELECT. This means that CP/M is aware that there is no C: drive on the system. Press return and after some disc activity the A> will reappear. You may select between A: and B: drives at any time when the A> or B> prompt appears (provided you have B: connected to the system). Selecting a drive in this manner makes the drive selected the default drive for all in-out disc operations. This simply means that if you save a program to disc it will be saved on the currently



active drive (the one with it's letter before the > sign).

This idea of a default drive is essential to master only if you have the B: drive attached, otherwise all in-out operations will take place on the A: drive (the only drive)

DIR *(parameters):* where parameters may be letter: (see above) or any combination of a filename and wildcards. The wildcards allowed are the asterisk (*) and the question mark (?). Here are some examples:-

Type the following and note the results:

```
DIR ?.?          (Return)
DIR *.*          (Return)
DIR *.?          (Return)
DIR *.BAS        (Return)
DIR *.???        (Return)
DIR A:           (Return)
```

Taken in order you will get the following results - no file; all files listed; no file; EX1. BAS & EX2. BAS; all files and finally all files again. Considered carefully, you can see that judicious use of the wildcards can greatly assist you when you are searching for files in any discs directory. You may even specify a wildcard in the middle of a file name such as EX?.BAS and using this as a DIR parameter you will get EX1.BAS and EX2.BAS listed. In short the wildcard "?" may be used in place of any one letter in a filename and the "*" may be used as any group of letters in filename. Note that when using the "*" any letters after it are ignored. If you have two drives connected you can get a directory listing of the disc in the other drive by including it's drive letter in the command (e.g. DIR B:, DIR A:).

ERA *(mandatory parameter):* where the parameter is any combination of wildcards and filename. Be careful!! Any file that is erased is not recoverable under ordinary circumstances!! Say for example that you were testing a program and it wrote to the disc a file that had the extension (the bit after the full stop) ".DAT".

After you had finished testing the program you may wish to reclaim the space on the disc by erasing all files that have been created by the program under test.

To do this you could ERA each one individually, but it would be quicker to use a wildcard of the form ERA*DAT which would erase all files with the extension of DAT. This can be hazardous if you have other files with that extension on that disk as it will erase those files also. You should always check which files are going to be erased by doing a DIR of the disc before erasing because erasure is permanent!! Be careful!! If you are

you are currently using the A: drive as the default you may erase on the B: drive by using ERR B:PARAMETER and vice-versa.

REN *(parameter) (separator) (parameter):* where the parameter is a filename, the separator is the equals sign (=) and the parameter is a filename. No wildcards are allowed and if you think about it, you will see that mass confusion would reign if you renamed a bunch of files to one name as CP/M would not know which file to access on demand. If you are currently logged onto the A: drive you may rename a file on the B: drive by using

```
REN B:NEWNAME=OLDNAME
```

That is you may rename on any disk drive from any disk drive but you may not specify a filename on to B: drive to be renamed on the A: drive or vice-versa. Rename does not transfer files across from one drive to another.

TYPE *(parameter):* where the parameter is a filename preceded by a disk drive letter code if the file is not on the currently logged disk drive. No wildcards are allowed. This command types A file to the screen (or terminal as it is called in CP/M). Strange results are guaranteed if you try to type any file other than one containing straight text. On your master disc is a file called DUMP.ASM. We will now type it to the screen using the type command. After the A> prompt type this:-

```
TYPE DUMP.ASM (Return)
```

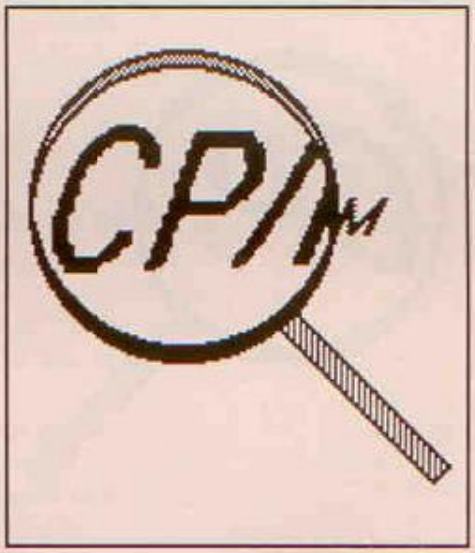
The text of that file scrolls up the screen at a fairly fast clip. Later we will see how we may halt the scrolling at any point and then restart it but for now it is enough to know that we can access files in this manner. As an exercise try the following

```
TYPE PIP.COM  (Return)
```

After the kaleidoscope effects have worn off reset the computer and then restart CP/M by using |CPM and Enter. That is why it is unwise to try to type a pure binary file as it can contain control codes that cause the computer to wander off into limbo land.

USER *(parameter):* where the parameter is a number in the range 0 to 15. If no parameter is present the current user number is displayed. The user number is used to partition discs into separate areas so that directories do not become cluttered with the so-called system files when displayed. On a single user computer like the Amstrad they have little value although if anyone was to attach a hard disk drive to their computer they would find the user number a great help in keeping their storage organised.

SAVE *(number) (parameter):* where the number is a count of pages



(256 bytes areas of memory starting at 0100 HEX) to be saved to the current disk drive under the filename parameter. This is a command that will be used by programmers to save their latest version of their masterpiece to disk. It is not a command that will be used in the maintenance of files or used by the non-programming user.

Those are the built-in commands of CP/M and with careful use and an understanding of each one file maintenance is greatly simplified. Each command is available from the A> prompt and will reward study.

The *transient commands* are those listed on your master disc copy with a file extension of "COM". COM denotes a command file (or program) that is loaded into memory and then executed. Do a DIR of your copy disk. Note all the files with the extension of COM. Each of these is actually a program that runs under CP/M and all of those supplied with the original CP/M disk are what are called utility programs. That simply means that they are there to make your job as computer operator easier. Try this at the A> prompt:-

```
STAT                (Return)
```

Assuming you were logged onto the A: drive the disc sprang to life and gave you information about the drives attached to your system.

What actually happened was this:- the memory resident portion of CP/M took what you had typed (STAT) and tested it to see if it was a built-in command. Finding that it wasn't it then looked at the current drive (the A: drive on single drive systems) for a program with the name STAT and the extension COM. Having found it, it then loaded this file into memory at 0100 HEX and then entered this file at that address. This file (or program) then did it's thing and ended up by returning to CP/M awaiting your next command. Now rename the file STAT.COM to STAT.BAS using the rename command as explained earlier. Now type (at the A>prompt):

```
STAT                (Return)
```

CP/M responds with STAT? and the A> prompt.

What happened was that CP/M could not find a file in the current disk with the name STAT and the extension of COM and so queried you on the command as entered. From this you can see that CP/M only executes files that have the extension COM. Now rename the file ROINTIME.DEM to ROINTIME.COM. and then type (at the A> prompt):

```
ROINTIME            (Return)
```

CP/M responds by loading the program and then we are in forty column mode with funny looking lettering and a barely discernible A> prompt. Do not attempt to use any commands but reset the computer and then re-invoke CP/M.

What happened? Well, CP/M found and loaded the program ROINTIME.COM and then jumped to location 0100 HEX and followed the code there to it's logical conclusion. But, since ROINTIME was a file written under AMSDOS it did not conform to CP/M's idea of a program written for it. This points out the difference between Amsdos and CP/M. While some (machine code) programs are interchangeable between the two you must not automatically assume that you can run Amsdos machine code programs under CP/M or COM files under Amsdos unless the programmer has taken special precautions.

To sum up so far, we now know that CP/M is a program designed to help you utilise the power of your disc drives. It is an operating system which has two types of commands, built-in and transient. Built-in commands are accessible immediately after the A> prompt while transient commands must be on the disc in one of your drives so that the command will work.

Now for a little technical information. CP/M consists of a number of different parts.

The Console Command Processor *(CCP)* is the portion that takes all that you have typed and tries to execute it.

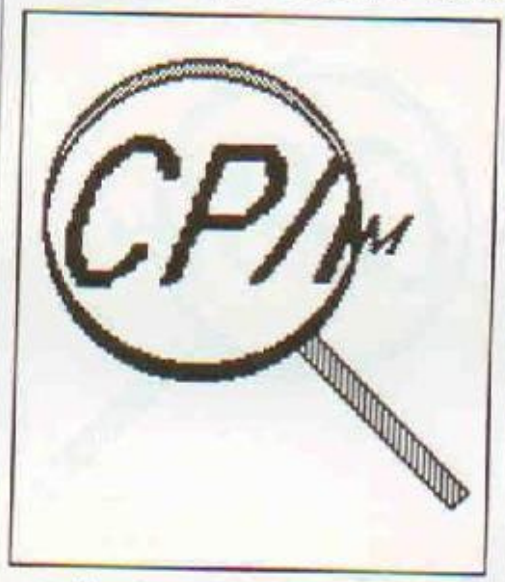
The Basic Disc Operating System *(BDOS)*. which incidentally has nothing to do with the language 'Basic', is the part that processes requests passed to it by the CCP at filename level or acts on what is known as 'service calls' to facilitate some aspect of disc access. Active ".COM" files (a .COM file is active when it is in memory as opposed to when it is on disc) usually use the BDOS to access discs as this then makes them independent of computer type and can then be run under almost any CP/M system.

The Basic In/Out System *(BIOS)* is the part that will actually do all the hardware accesses, perform the writing to disc, screen etc. This is the part that is modified for each different computer usually by the person who makes the computer and is the only part to change.

Low memory is the area in the computer's memory from address 0000 hex to 00FF hex and contains system parameters in a known configuration so that all .COM programs know where to find them.

The Transient Program Area *(TPA)* is the part of the computer's memory that starts at 0100 hex and continues until the top of available memory. This is from where all .COM files are loaded and executed.

The CCP also recognises some



control codes to ease the operators task of conversing with the system. These are all accessed by holding down the CTRL key and the letter key together and are:

CTRL C This causes the CCP to be reloaded into (high) memory. It can also cancel active .COM files and is considered the usual way to end such programs.

CTRL E This does a carriage return on the screen, but doesn't send it to the CCP. Useful for entering long command lines.

CTRL H A destructive back-space which wipes out the previous character.

CTRL I Generates a Tab (the assumption is 8 spaces for Tab).

CTRL J Gives a line feed and terminates input from the console (the keyboard).

CTRL M Gives a carriage return and terminates input from the console.

CTRL P This is a printer toggle switch. Enter it once and all that appears on the screen will be echoed to the printer (assuming it is on). Press it again and the printer stops.

CTRL R Erases and also retypes the current CP/M command line. This is a hangover from the days when teletypes were used as consoles and had no backspace key.

This could have led to unreadable command lines.

CTRL S Suspends the console output until another key is pressed. *Use TYPE to put the file DUMP.ASM on the screen. If you are quick enough, using CTRL S will suspend the output.*

CTRL U This is another hangover from the TTY days. It ignores the current command line but does not erase it and then moves down to the next line ready for the next command.

CTRL X Erases the current line and returns to the start of the line.

CTRL Z Terminates the console input and also acts as a separator in some .COM files (notably ED.COM).

These control codes all work on the Amstrad but CTRL Z can have some funny side effects if used at the A> prompt.

You can see from this that there is quite a wealth of detail yet to be uncovered by the CP/M explorer.

In between this article and the next, do not be frightened to play around with the disc we have made as it is quite easy to make another if you wipe out this one.

In the next article we are going to go deeper into the transient commands and also the way in which CP/M communicates with those devices, such as printers, disc drives and consoles, which are attached to your computer.

This is the first of a series of articles by Mark Godden covering the CPC6128 and should shed some light for new and existing owners.

Since one of the major differences between the CPC6128 and its predecessors is the Operating System (which is called CP/M Plus), I think that it would be sensible to break this first article into two sections, looking firstly at the 6128 itself with a demonstration program then the operating system supplied with the machine.

Elsewhere in this magazine (Page 5) is a new series on CP/M, so I will be making the assumption that the reader has some prior knowledge and is familiar with such things as Drives and Transient Programs.

The CPC6128

This machine is basically a CPC664 with the exception of having 128k of memory with the added ability of running CP/M Plus as its second Operating System (OS for short). The first, or native operating system is AMSDOS.

The Central Processing Unit (CPU) being a Z80A can only access 64k of memory addresses at a time, so the second 64k (making up the total 128k) is Bank Switched, a Bank referring to the second 64k block as a whole unit. So with the use of a small program supplied on your master disc, you can access the second Bank from your programs and Basic giving you 64k extra storage. The program that allows you to do this is called BANKMAN.BAS which loads a special binary file and enables the use of extra commands from Basic through RSX's (Resident System Extensions), but more on these later.

The program that follows will give a demonstration of Bank switching 16k blocks of memory (one complete screen) and provides an on-screen catalogue of both sides of two discs. This function is not available on the 464 or 664. It is full of REM statements and I hope will be of some use to owners. You'll find that because it uses the |SCREENCOPY ,n,n command which copies the 16k block to the 'current screen' 16k block rather than swap, you will always have a copy of the CAT in memory even after typing NEW! Don't forget that before running your debugged and saved program, run BANKMAN.BAS.

```
1    '*****************************************
  ************************
2    '*     BANKCAT.BAS                  WRITTEN
  BY M.C.Godden 30/01/86 *
3    '*                                  *
4    '*       NOTE : This program only runs
  on the CPC 6128        *
5    '*****************************************
  ************************
6    '
10   ' This program takes a catalogue of two
   disk's both sides
20   ' and stores the screen image in the se
  cond 64k bank by
30   ' using the command |CREENCOPY,n,n you
  can have a instant
40   ' catalogue of the disk's that your cur
  rently using.
50   '
60   ' Run BANKMAN.BAS first.
70   ' Then select the disk's you will be us
  ing , run this program
80   ' and follow the screen prompts.
90   '
98   ' Set up variables
99   '
100  MODE 1:side=0:scrnno=2:number=0:repeat=0
  :DIM prompt$(4)
101  REM read data into array
102  FOR readdata = 0 TO 3:READ prompt$(readd
  ata):NEXT readdata
105  REM Main procedure
110  LOCATE 3,2:PRINT prompt$(number):REM Dis
  play prompt
115  IF repeat=2 THEN LET side=0:REM Reset si
  de counter to 0
120  GOSUB 1000:REM Main Routine
130  IF repeat<4 THEN GOTO 110:REM Jump back
  to loop
140  '
150  ' Demo routine allows a demo of what thi
  s program has done
160  ' this routine calls a delay routine at
  line 2000.
170  '
180  CLS:LOCATE 6,6:INPUT "RUN DEMO Y/N ";dem
  o$
190  IF demo$="Y" OR demo$="y" THEN GOTO 200:
  REM Run demo
195  LOCATE 10,1:END
200  REM Demo Routine
210  LET scrnno=2
220  |SCREENCOPY,1,scrnno
230  GOSUB 2000:REM Delay Routine
240  IF scrnno<6 THEN GOTO 220
```

```
 250  LOCATE 10,1:END
1000 '
1001 'The following routine gets the disk labe
     l and displays the continue
1002 'message then waits for a key to be press
     ed , once pressed it increments
1003 'the variable (SIDE) by 1 ,clears the scr
     ren does a catalogue of the disk
1004 'copies the screen to the second bank. Af
     ter that is sets the repeat flag
1005 'increments the screen number variable (S
     CRNNO) by 1 then returns for the
1006 'next disk.
1007 '&BB18 is the WAIT FOR KEY ROUTINE.
1008 '
1010 LOCATE 3,4:INPUT "DISK LABEL ";disklab$
1020 LOCATE 3,6:PRINT "Press any key to contin
     ue":CALL &BB18
1030 LET side=side+1
1040 CLS:LOCATE 1,1:PRINT "Disk Label ";dickla
     b$;"     Side    ";side
1050 CAT
1060 |SCREENCOPY,scrnno,1:REM Copy screen cont
     ents to second bank
1070 LET scrnno=scrnno+1:REM Increment screen
     number block
1080 LET repeat=repeat+1:REM Increment repeat
     counter
1081 LET number=number+1:REM Increment prompt
     ARRAY
1090 CLS:RETURN
2000 'Delay Routine
2001 '
2005 REM THATS ALL FOLKS
2010 FOR delay=0 TO 2000:NEXT delay
2020 LET scrnno=scrnno+1:REM Increment Scrren
     counter
2030 RETURN
3000 'Data
3001 DATA "PLACE FIRST DISK IN DRIVE (SIDE 1)"
3002 DATA "PLACE FIRST DISK IN DRIVE (SIDE 2)"
3003 DATA "PLACE SECOND DISK IN DRIVE (SIDE 1)
     "
3004 DATA "PLACE SECOND DISK IN DRIVE (SIDE 2)
     "
```

CP/M Plus

The big plus (excuse the pun) with this OS is the fact that it has a 61k Transient Program area (TPA) compared with only 38k on the 464 or 664. So running CP/M Plus should give you access to a larger amount of CP/M software.

CP/M Plus is compatible with CP/M 2.2 discs - in other words if you load/boot CP/M Plus it can read discs formatted and written to by CP/M 2.2.

Which is better - CP/M 2.2 or CP/M Plus? Well, it's not really fair to compare the two even though there is a CP/M Plus version that runs on a 64k machine. Its normal operating environment is 128k or bigger and CP/M 2.2 is already well proven on the 464 and 664. However, one problem with CP/M 2.2 on the 64k machines occurred while switching to B: drive when your configuration was not currently a two drive system. It forced an error message to be displayed advising that B: drive was missing. This problem is not present with CP/M Plus. When you switch to B: a scrolling message on the status line at the bottom of the screen states INSERT B: DRIVE DISC AND PRESS ANY KEY.

On the master disc is a program called DISCKIT3.COM which provides a menu-driven utility for formatting, copying and verifying discs. There is also one available for CP/M 2.2 but is slower because it uses less memory. These programs allow you to format a disc while copying, detecting the type of format on the original and adjusting the format on the copy accordingly.

CP/M Plus uses the whole 128k as such, and AMSDOS has been designed to interface directly to its standards. But, as mentioned in the manual, PASSWORD, DISC LABELLING and DATE stamping are not supported by CP/M 2.2 or AMSDOS.

There are a lot of extra utilities supplied with CP/M Plus and I will take a look at these next month. As we progress I will endeavour to explain the differences and advantages between CP/M Plus and CP/M 2.2. But for now, be aware that CP/M Plus is not another OS to learn, rather it is a logical progression from 2.2 that Amstrad have taken advantage of in the release of the 6128.

Trivia

There isn't any with this first article, but the idea is to answer questions and to point out any problems (and their possible solutions) in running software on the 6128. Readers are welcome to divulge their discoveries.

HAPPY COMPUTING !!

SIMPLE!

All correspondence should be addressed to
The Editor,
The Amstrad User
Suite 1, 33 The Centreway
Mt. Waverley, Victoria 3149

Towers of Logo

from Jeff Hughes

Logo is a computer language which, like Pascal, is based on the use of procedures and supports recursion. It is interesting then, to try to implement some of the best known recursive algorithms in LOGO. One of the most popular is the "Towers of Hanoi".

The mathematical puzzle is based on an ancient legend, where a sect of monks in Tibet were set to solve the puzzle on the day of creation. When they finally achieve the solution the world would end in a clap of thunder. The game consists of a set of disks (64 in the monks' puzzle) of descending size on peg one. There are two other pegs, and the disks must be moved to peg three, one at a time, according to the rules:

1. Only the top disk on any peg can be moved, from its peg to any other peg.
2. A larger peg cannot be placed on top of a smaller one.

The basic algorithm (in Pascal-type code) is as follows:

```
hanoi (n,from,use,to)
if n<>0 then
hanoi (n-1,from,to,use)
print move(from,to)
hanoi(n-1,use,from,to)
```

Using this algorithm it takes (2^n - 1) moves to move n disks (for n=64, if the monks make a move a second it will take 565,000,000,000 years till the universe ends. We still have a little time yet!).

Listing One shows the implementation of this algorithm in LOGO. The third line is the code necessary to print out the moves. However, since LOGO is a graphics language it would seem more interesting to display the movement of the disks.

Try hanoi 4 1 2 3 for the moves needed to move 4 disks from peg 1 to peg 3.

The second listing contains the procedures to implement the graphics version of the program. As well as showing LOGO's use of recursive procedures and graphics, the program also demonstrates the list handling features of LOGO. The program uses lists to keep track of the disks on each peg.

The procedures are:

`LOCATE` - sets the turtle to a given position and heading. Note that `'setpos'` will not accept variables as input, for example `setpos [:x :y]` is not allowed, but *will* accept lists as input as follows:

```
make "l list :x :y
pu setpos :l pd
seth :ang
```

This code could be used as an alternative to `'locate'`. This version also has the advantage that, if `'pu'` is deleted, the procedure will draw a line to a position specified by (x,y).

`PEG` - draws a peg

`DRAWPEGS` - draws the three pegs using `PEG`

`DISK` - draws a disk of size `'w'`

`DRAWSTACK` - draws `n` disks, of reducing size, on the first peg.

`MAKELIST` - sets up the list [1 2 ..... n] for keeping track of the disks

`DECSTACK` - erases a disk of size `i` from peg `p` and adjusts the list of disks on peg `p`

INCSTACK - draws a disk of size i on peg p and adjusts the list of disks for peg p

SETINDEX - initialises the indices if and it (the number of disks on 'from' and 'to') and the lists fl and tl (the list of disks on 'from' and 'to')

ALTINDEX - adjusts the indices n1, n2 and n3 (the number of disks on 1, 2 and 3)

HANOI - implements the algorithm, updates indices and lists and draws the movement of disks

RUNHANOI - initialises the graphics by drawing the pegs and stacks of disks, initialises the indices n1, n2 and n3 and lists l1, l2 and l3 and then calls the hanoi procedure.

The graphics can handle up to six disks (this takes 2^6-1, ie 63 moves). DR LOGO graphics are *very* slow, but if desired the program could handle larger values of n by adjusting the size of the disks. For a reasonable demonstration try

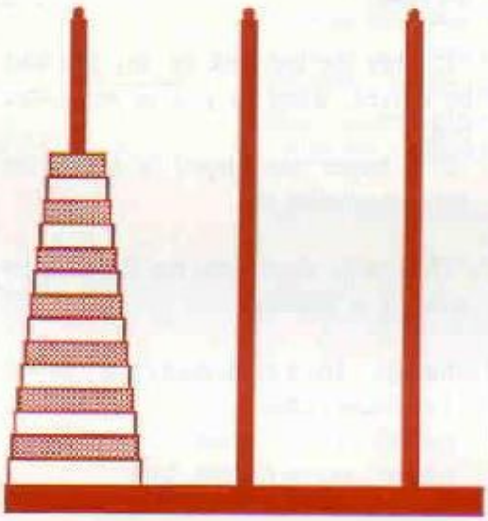
```
HT RUNHANOI 5 1 2 3.
```

LISTING 1

```
to hanoi :n :f :u :t
if :n = 0 [stop]
hanoi :n - 1 :f :t :u
type [from] type [_] type
:f type [to]
type [_] type :t pr []
hanoi :n - 1 :u :f :t
end
```

LISTING 2

```
to locate :x :y :ang
pu setpos [0 0] seth 0 fd
:y rt 90 fd :x
seth :ang pd
end
to peg
fd 50 lt 90 fd 70 bk 70
rt 90 fd 50
end
to drawpegs
locate -250 0 90 peg
locate -50 0 90 peg
locate 150 0 90 peg
end
to disk :w
seth 0
repeat 2 [fd 10 rt 90 fd
100 - 20 * :w
rt 90]
end
to drawstack :n
seth 0 make "w 0
repeat :n [disk :w make
"w :w + 1 pu
seth 0 fd 10 rt 90 fd 10
pd]
end
to makelist :n
make "i 1 make "l 0
repeat :n - 1 [make "l
fput :i :l make "i :i +
1]
end
```



```
to decstack :p :i
locate -250 + (:p - 1) *
200 0 90
make "d first :fl
make "fl bf :fl
pu fd 10 * :d lt 90 fd 10
* :i pd
pe fd 10 rt 90 fd 100 -20
* :d rt 90
fd 10 pd
end
to incstack :p :i
locate -250 + (:p - 1) *
200 0 90
make "tl fput :d :tl
pu fd 10 * :d lt 90 fd 10
* :it pd
fd 10 rt 90 fd 100 - 20 *
:d rt 90 fd 10
end
to setindex
if :f = 1 [make "if :n1
make "fl :l1]
if :f = 2 [make "if :n2
make "fl :l2]
if :f = 3 [make "if :n3
make "fl :l3]
if :t = 1 [make "it :n1
make "tl :l1]
if :t = 2 [make "it :n2
make "tl :l2]
if :t = 3 [make "it :n3
make "tl :l3]
end
to altindex
if :f = 1 [make "n1 :n1 -
1 make "l1 :fl]
if :f = 2 [make "n2 :n2 -
1 make "l2 :fl]
if :f = 3 [make "n3 :n3 -
1 make "l3 :fl]
if :t = 1 [make "n1 :n1 +
1 make "l1 :tl]
if :t = 2 [make "n2 :n2 +
1 make "l2 :tl]
if :t = 3 [make "n3 :n3 +
1 make "l3 :tl]
end
to hanoi :n :f :u :t
if :n = 0 [stop]
hanoi :n - 1 :f :t :u
setindex
decstack :f :if - 1
incstack :t :it - 1
altindex
hanoi :n - 1 :u :f :t
end
to runhanoi :n
locate -250 0 90
drawpegs
locate -250 0 90
drawstack :n
make "f 1 make "u 2 make
"t 3
make "n1 :n make "n2 0
make "n3 0
makelist :n
make "l1 :l make "l2 0
make "l3 0
hanoi :n 1 2 3
end
```

Software Reviews

DALEY THOMPSON'S DECATHLON
Reviewed by Darren Robinson

Ready, Set...GO!! You'll need to be fit to take on this gruelling event. As the title suggests, there are ten events to compete in and the object in each is to beat the set qualifying time or distance. This is achieved by wiggling your joystick as fast as you can; in the field events you must then hit the fire button. (It's not essential to have a joystick but after a few games you'll start feeling sorry for the keyboard!)

Those who've played the coin-op 'Hyper Olympic' will find the theme familiar. Hard to beat? Not really, a little practice and a lot of aggression and you should be completing all ten events after a while.

The graphics in DTD are quite okay but it is the animation which is a letdown - no matter what the speed bar shows, Daley always seems to jog along at his own pace. The movement of the long jump official is very jerky; it may have been better to leave him out altogether.

Documentation to me was not very helpful, in fact it omits to say which keys to push! The better you become at this game the more it's going to hurt! There are no easy sections or learner's phase or dem screen - it's just head down and go for it. It's a pity there isn't a 2 player option as this sort of game is best played against your friends.

Another drawback is that once you do complete all ten events, the qualifying times don't get any harder. It just becomes a matter of endurance thus diminishing the long term appeal of the game.

Summary: Worth getting for its fun, rather than technical value.

RATINGS: (out of 8)	
Ease of use	7
Speed	3
Entertainment value	8
Documentation	2
Originality	4
Use of graphics	4
Ability to hold interest	7
TAU INDEX	**62%**

AIRWOLF
Reviewed by James Gallagher

To misquote an old song, Airwolf by Elite Systems is a case of "nice graphics, shame about the game." There's a pretty loading screen, big helicopter cockpit style score readouts, catchy theme music and realistic chopper blade sounds, not to mention the super 'copter itself, which is suitably sleek and deadly. What you don't get is playability - it's simply much too hard. The instructions are also rather vague.

Before I go any further, let me explain that the game is based on an American television series (aren't they all nowadays?) which is yet to be seen in Australia. The hero is one String Fellow Hawke, and ex-Vietnam chopper pilot (sounds more like one of the Fantastic Five) who just happens to be the only living being in the entire Universe who can pilot the U.S.'s latest high-tech helicopter (hands up who's already switched over to the ABC!) Thus, in the game you have the dubious distinction of being the only person in the entire Universe who can save the five important U.S. scientists being held hostage deep in a subterranean base beneath the scorching Arizona desert. To do this you must destroy the defence control boxes rather clumsily scattered about the caverns and this is where the trouble begins.

Complaint 1: - you are racing against a timer, which counts down far too quickly. For some unexplained reason your chopper explodes when it reaches zero.

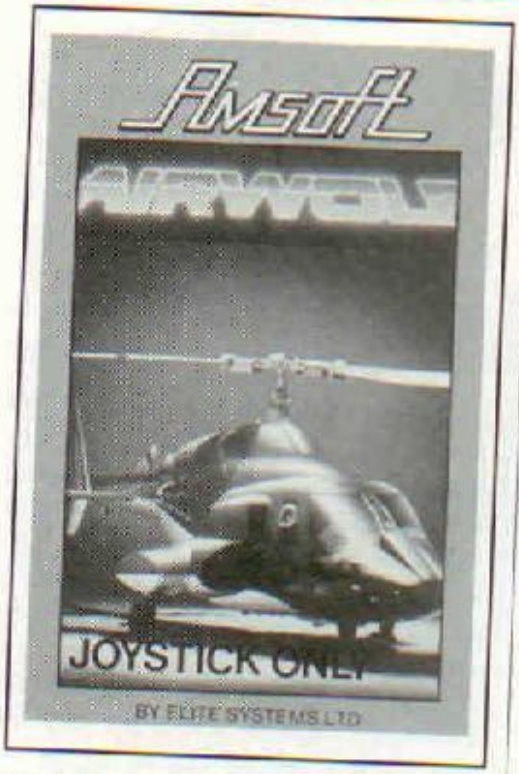
Complaint 2: - manoeuvring space is limited and your chopper reacts *very* quickly to the controls - enough said.

Complaint 3: - you don't have "lives" as in other games, but little shields. You lose one every time you brush up against the walls or some other obstacle. Losing them all takes a ridiculously short time. Be prepared for a game that lasts for two seconds - *LITERALLY*.

Complaint 4: - your chopper has some very basic problems for such a sophisticated piece of equipment, for example not being able to hover (and they call this a HELICOPTER!) and a surprising lack of armaments - a single, rather feeble cannon is all you have.

Complaint 5: - when you actually manage to bullseye a control box nothing happens except that a little blue square disappears from the lid. The significance of this quite escapes me.

Complaint 6: - the game is joystick only, a curious move as only a few extra lines of code are needed to test the keyboard. The instructions neglect to



mention how to start the game (pressing the fire button) or that the space bar pauses the game. The scoring is not explained nor are the workings of the time which is labelled "BONUS".

These might seem like minor quibbles but they really do ruin what could have been a very good game. I consider myself a hardened arcade game freak so when I say that it's fast, I mean *FAST* - adventure game enthusiasts won't have a snowball's chance in h***. I soon became tired of losing life after life after life, even when I discovered a way of cheating (hold down the space bar while playing - when you release the joystick all action will freeze).

The redeeming feature of the game is it's graphics. The top third of the screen contains the score, high score and bonus timer displays, all done in big, digital watch type numerals. The bottom two lines or so of the screen contain the shields, of which there are six. The space between these regions alternates between the AIRWOLF-ELITE logo while the game is not played, and a section of the cavern. This scrolls in the appropriate direction when the chopper reaches one of the sides.

The cavern consists of a number of chambers of various sizes, each containing an assortment of defence devices, including the control boxes. The chambers are separated by thick blue columns which must be blasted out of the way before you can proceed.

I don't consider myself to be a moron but Airwolf left me quite bewildered. It could be that the game contains an as-yet unremoved bug, or that the instructions omitted some crucial detail. Technically it's very good, with smooth, colourful graphics, nice sound and a good screen layout. Despite the difficulties I won't be deterred - some day I'll get through to those poor guys. Whether or not they'll still be alive then is another matter....

RATINGS: (out of 8)	
Ease of use	7
Speed	7
Entertainment value	6
Documentation	3
Originality	5
Use of Graphics	7
Ability to hold interest	7
TAU INDEX	**75%**

MYRDDIN FLIGHT SIMULATOR
Reviewed by Philip Crosby

If you're tired of destroying enemy starships and bored with finding your way through adventure games, then maybe the Myrddin Flight Simulator is for you. Although not strictly game, the program is a very clever and sophisticated piece of software requiring concentration, dexterity, anticipation and good reflexes, and, if you happen to be a pilot, flight simulator will test your skill under "instrument" conditions with surprising realism.

After loading the program, you are faced with a view out of the front of a cockpit looking down the runway with the engine idling. The lower half of the screen shows the instruments and switches required for flight and upon opening the throttle, you appear t move down the runway with increasin speed. Pulling back the joystick lift the aeroplane off the runway an continues the climb to the desire altitude. Once airborne you ar absolutely free to "fly" around th landscape, over towers, fields an coloured landmarks, at any height speed or direction provided the pre-se limits are not exceeded. A map is provided showing the latitude and longitude of each landmark and the aeroplanes' position is displayed on the instrument panel.

The landing phase is where your skill must be finely tuned to position the aeroplane at the correct height, speed and rate of descent to facilitate a smooth arrival at one of the 3 run-ways available. Every landing is different and only the correct setting up and constant re-appraisal of altitude and position will avoid a crash.

There are 15 performance levels which effectively vary the aircrafts response to the joystick movements. Level 1 is good for learning and practice landings although I found levels 3-5 better for realism and a more immediate response to the controls. Levels 10-15 are really only for aerobatics and fancy flying. (Keeping your airspeed between 80-120kts will also help to practice your manouvering).

Unfortunately, there is one error in the program which gives a stall speed of 45 kts with flaps up and 65 kts with flaps down. These should be vice-versa giving a lower stall speed with flaps lowered enabling a slower landing speed. Some labelled sticky-tape over the screen corrected the problem but the drag effect of the flap is still a nuisance, and the writers should rectify this mistake as soon as possible. The program makes excellent use of the Amstrad graphics and with a bit of imagination you'll be surprised at how realistic it all becomes. Don't expect to be an "ace" straight away because, like the real thing, it can take many hours to reach the "solo" standard and complete that first safe landing.

(No Ratings provided).

The Learning Centre

An Introduction to Music - Part Six

by Peter Campbell

As promised last month, this is the final article in this series on programming music on your Amstrad. I trust that you have learned something useful along the way. I certainly have, not the least from the book that I mentioned last month, "Making Music on the Amstrad CPC464 and 664" by Ian Waugh.

Envelopes the 'Hard' Way

One thing that had puzzled me, despite reference to the User Instructions and the Firmware Specification, was how to use the hardware envelopes that are a feature of the AY-3-8912 sound chip. Mr. Waugh makes it quite clear that it is really a simple process to access the sound chip's inbuilt volume envelopes. He also reveals a way the hardware tone envelope capability can be utilised. The secret is hereby nicknamed 'Iggles'. (Surely you have heard of him - two times two iggles four!) Jesting aside, a simple equals sign is the key to adding hardware envelopes to your music program.

8-15 Is Not The Time

The hardware envelopes are numbered 0 to 15, but 0 to 7 only duplicate some of the main envelopes, which are numbered 8 to 15. The envelopes do not include a pause time, so this must be added by means of a software segment. Thus hardware envelope number 9 could be programmed:

```
135 ENV 2,=9,5000,2,0,
    duration
```

The parameter, 5000, is called the envelope period and is measured in 128ths of a second (i.e. 1280 would be 10 seconds).

If that envelope is inserted into "Silent Night" (for three voices), it has the effect of changing the continuous clarinet-like sound to one much more bell-like. You will need to add (volume envelope number) ',2' to line 140 and change (volume envelope number) '1' to '2' in line 200. You will also need to change the zeroes in the last two groups of data to, say, 125. (Don't make the duration longer than 127, though, or you will get a syntax error). I found the effect better when 'octave' was altered to '2' in line 90.

Getting an Earful of Hardware

The best way to understand the effects that can be achieved with the hardware envelopes is to listen to them. Try this short program:

```
10  MODE 1
20  FOR he=8 TO 15
30  FOR i=1 TO 11
40  READ ep
50  CLS:PRINT "HARDWARE
    ENVELOPE NUMBER (he):";he
60  LOCATE 1,3:PRINT
    "ENVELOPE PERIOD (ep)
    :";ep
70 LOCATE 1,7:PRINT"ENV 1,
    =";he;",";ep;", 5, 0,
    100"
80  LOCATE 1,9:PRINT"SOUND
    1, 239, 0, 0, 1"
90  ENV 1,=he,ep,5,0,100
100 SOUND 1,239,0,0,1
110 another$=INKEY$:IF
    another$="" THEN 110
120 NEXT i:RESTORE:NEXT he
130 DATA 10,15,25,50,100,200,
    300,1000,2000,4000,5000
```

Allow five seconds before pressing a key to listen to the next envelope; some work better with long envelope periods and some don't appear to work with short ones.

A Different Tone

Whilst hardware volume envelope modifies volume in a manner similar to that of a software volume envelope, a hardware tone envelope is quite

different from its software 'cousin'. It takes the form: =TP,PT where TP is the tone period (pitch) of a note and PT is the pause time (duration) of the note. Hardware and software sections can be mixed, but one use of a hardware tone envelope is to hold the notes for a short tune e.g. an effect in a game, such as this one:

```
10  ENT -1,-478,10,-359,
    10=358,10,-319,10
20  SOUND 1,0,90,7,0,1
```

Vibrato

That TV show where they pontificated about such things may no longer be with us, but 'vibrato' certainly is. In a musical vibrato the pitch rises and falls regularly in a sine wave pattern above and below the pitch of the note. The variation should be kept to less than a semitone, unless a special effect is desired. As a 'rule of thumb, Mr. Waugh suggests:

```
ENT -1,X,D,S,2*X,-D,S,X,D,S
```

where D represents the depth of the vibrato and S the speed. X (the note you start from) will have an effect on both these factors and thus this is only a starting point for your experiments. To produce a sine wave type of vibrato, you should make the middle section twice that of the ends, hence the factor of 2.

A Trilling Time

It is impossible to produce a vibrato on a piano. As we know, its notes are a semitone apart and our vibrato would fall in the cracks between the notes. The best we can do is to alternate rapidly between two adjacent notes, producing a trill.

If we want to include a trilling flute at the beginning of a martial piece of music, we can instruct the computer to play the two notes just as we would any other notes. E.g., using our newly-acquired hardware tone envelope:

```
ENT -1,=53,8,=67,8,=53,8,=67
,8
SOUND 1,0,240,0,1
```

REVERBERATION...
reverberation.. beration.. ation..

Mr. Waugh feels that echo and reverberation are two of the most overused effects on a synthesiser, but they have their uses. An echo is a straight-forward reflection of a sound, separated from the original by a time lag of at least one-tenth of a second. Reverberation describes the multiple reflections produced in a confined space. The latter is not strictly possible on an Amstrad, but we can define an envelope that gives something of the effect.

```
10  ENV 1,1,15,20,8,7,20,1,
    -7,20,1,5,20,2,-2,20
20  SOUND 1,478,-1,0,1
```

Once volume reaches 15, the next step takes it to zero, producing a wrap around. If you sketch the envelope out on a piece of squared paper, wrapping it around as it reaches the maximum setting you will see how the reverberation is created by the above lines.

A Job in the Chorus

Why do two violins playing together sound different from a solo instrument? Despite the musicians carefully tuning, the effect is caused by the two musicians playing at a slightly different pitch. The difference is quite small, but it tells our ears what we are listening to.

This effect is easily produced. Just add a second SOUND line to any piece of music you have handy:

```
10  SOUND 1,pitch,duration,
    volume
20  SOUND 2,pitch-1,duration,
    volume
```

Instead of subtracting 1, try subtracting values up to 4 and select the one you prefer. You could even try the effect on all three channels by adding a third sound command:

```
30  SOUND 3,pitch-2,duration,
    volume
```

It Sounds Different To Me

I have mentioned before the importance of add-on hardware in getting the best out of the 'stereo' capabilities of the Amstrad, but before concluding this series, I cannot resist mentioning it again.

The internal speaker severely limits the sound. By taking the sound output from the socket on the back to a suitable amplifier and speakers, both music and sound effects are considerably enhanced. So much so that sound effects may even have to be rewritten to get back to the sound you had in mind. Such is the difference better speakers make!

Practice Makes Perfect

Practice makes perfect so they say. Well one thing we can do with our earlier listings is to produce a program that helps us practise music reading. You have, no doubt, kept a copy of the merged programs which appeared in The Learning Centre in issues 9 and 10 (October and November 1985). The following procedure relates to that merged file created in issue 10 (see Page 19) and should be followed closely.

LOAD "*The merged program name*"
DELETE 1 - 999
DELETE 1390 - 1600
DELETE 1740 - 1990
DELETE 2330 - 2610
Save the remainder as MLIST4 using the ASCII (,a) option, namely

SAVE "MLIST4",a

Now type in the lines from List 5 following this article and save them as MLIST5.

Merge them with MLIST4 and when you RUN the combined program you have a piece of music handy. Choose either the treble or bass clef and try to identify the notes in the piece of music (ignoring sharps and flats). If you think that the note is, say, a 'c' and it is in the lower part of the clef (or beneath it on a leger line), just touch 'c' on your keyboard and 'c' will be drawn on the

stave, enabling you to check your answer. If the note is in the upper part of the clef, then press CAPS LOCK first. Note that CAPS LOCK is automatically reset after every note.

I haven't included provision to play the note, but this could easily be done, working from the varible 'note' and using the formulae for frequency and tone period. You will have to adjust the value of 'note' so that 'c' is 1 in the octave selected by the use (or non-use) of CAPS LOCK and the cursor keys. I have even left in the complete note drawing routine so that you can have notes of different length!

The keyboard from our earlier listings can be incorporated into a synthesiser program. Again, referring to the original merged file we need to get rid of the extraneous bits:

LOAD "*The merged program name*"
DELETE 1 - 1020
DELETE 1060 - 1380
DELETE 1610 -

Not much left, is there? Save it as MLIST6, again using the ASCII option to ensure that you can merge the listing without difficulty.

Now type in the lines from List 7 following this article and save it as MLIST7. Load MLIST6 and then merge MLIST7 to create your 'Amstrad Synthesiser'. You can SAVE the result as whatever name you like, say MLIST8? Here's how it works:

Lines 1030-1060 amend the original keyboard listing and substitute the actual keys you will press for the note names (A# etc.).

Lines 6020 restores normal keyboard operation and normal screen colours when ESC is pressed twice - see line 5020

The sub-routine starting at line 5720, initialises the various envelopes and the variables, including one called 'keystr$', which contains a list of the acceptable keys - note the space after 'b'. CHR$(16) is the CLR key, CHR$(&F1) are the cursor arrows for 'up' and 'down' respectively, while CHR$(16) is the TAB key. The routine also contains the lines which complete the screen layout.

The remaining routines do what their names suggest.

To use the synthesiser, play the tune on the keys '1' to 'CLR' and 'Q' to '['.

'Z' to 'B' and 'M' to '\' select the volume and tone envelopes, the space bar toggles the chorus effect, the TAB key toggles the 'harmony' effect and the up and down cursor keys change the octave setting.

Our generous editor has offered a prize for the best 'orchestration' of a piece of music, applying the principles learned during this series. Why not give it a try? For example, for a military piece, you could open with a trilling 'flute' over the deeper instruments. After the introduction, a chorus of 'trumpets' could take up the air, and so on. Good luck!

LIST 5

(Tape subscribers please note that MLIST45 contains the merged version of MLIST4 and 5).

```
   10 GOSUB 1630:GOSUB 1030:GOSUB 5000
            'AMENDED
   20 LOCATE 4,5:PRINT"Do you want to
      try again?              "
   30 LOCATE 4,7:PRINT SPC(32)
   40 WHILE INKEY(43)=-1 AND INKEY(46)
      =-1:WEND
   50 IF INKEY(43)=0 THEN RUN ELSE 802
      0
 1040 :
 5000 '
 5010 REM ******* Main Routine *******
 5020 '
 5030 e$="efgabcdEFGABCD   abcdefgABCD
      EFG":x=82:treble=-1:bass=-1
 5040 WINDOW 1,40,15,25:PEN 1:MOVE 0,0
      :PLOTR -2,-2,3
 5050 WHILE NOT numberofnotes=14
 5060 IF treble=0 OR bass=0 THEN 5130
 5070 CLS:LOCATE 4,5:PRINT "Press  ";C
      HR$(1);CHR$(11);"  for Treble Cl
      ef"
 5080 LOCATE 4,7:PRINT "   or  ";CHR$(
      1);CHR$(10);"  for Bass Clef"
 5090 WHILE INKEY(0)=-1 AND INKEY(2)=-
      1:WEND
 5100 IF INKEY(0)=-1 THEN treble=-1 EL
      SE treble=0
 5110 IF INKEY(2)=-1 THEN bass=-1 ELSE
       bass=0
 5120 WHILE NOT INKEY$="":WEND
 5130 POKE 46312,0                'CA
      PS LOCK off
 5140 IF bass=0 THEN e$=LEFT$(e$,14) E
      LSE e$=RIGHT$(e$,17)
 5150 CLS:LOCATE 4,5:PRINT"Press CAPS
      LOCK for upper octave"
 5160 LOCATE 4,7:PRINT"Then press name
       of note required"
 5170 note$=INKEY$
 5180 IF note$="" THEN 5170
 5190 IF (ASC(note$)<65 OR ASC(note$)>
      71) AND (ASC(note$)<97 OR ASC(no
      te$)>103)        THEN note$="":GO
      TO 5170
 5200 numberofnotes=numberofnotes+1
 5210 note=INSTR(e$,note$)
 5220 ON note GOSUB 6020,6050,6080,611
      0,6140,6170,6200,6230,6260,6290,
      6320,6350,       6380,6410,6440,6
      470,6500
 5230 WEND
 5240 RETURN
```

```
5250 '
6000 REM ***** Position Note *****
6010 '
6020 x=x+36:y=176
6030 GOSUB 7020:GOSUB 2020:GOSUB 7080
6040 RETURN
6050 x=x+36:y=185
6060 GOSUB 7020:GOSUB 2020:GOSUB 7080
6070 RETURN
6080 x=x+36:y=194
6090 GOSUB 7020:GOSUB 2020:GOSUB 7080
6100 RETURN
6110 x=x+36:y=203
6120 GOSUB 7020:GOSUB 2020:GOSUB 7080
6130 RETURN
6140 x=x+36:y=212
6150 GOSUB 7020:GOSUB 2020:GOSUB 7080
6160 RETURN
6170 x=x+36:y=221
6180 GOSUB 7020:GOSUB 2020:GOSUB 7080
6190 RETURN
6200 x=x+36:y=230
6210 GOSUB 7020:GOSUB 2020:GOSUB 7080
6220 RETURN
6230 x=x+36:y=239
6240 GOSUB 7020:GOSUB 2020:GOSUB 7080
6250 RETURN
6260 x=x+36:y=248
6270 GOSUB 7020:GOSUB 2020:GOSUB 7080
6280 RETURN
6290 x=x+36:y=257
6300 GOSUB 7020:GOSUB 2020:GOSUB 7080
6310 RETURN
6320 x=x+36:y=266
6330 GOSUB 7020:GOSUB 2020:GOSUB 7080
6340 RETURN
6350 x=x+36:y=275
6360 GOSUB 7020:GOSUB 2020:GOSUB 7080
6370 RETURN
6380 x=x+36:y=284
6390 GOSUB 7020:GOSUB 2020:GOSUB 7080
6400 RETURN
6410 x=x+36:y=293
6420 GOSUB 7020:GOSUB 2020:GOSUB 7080
6430 RETURN
6440 x=x+36:y=302
6450 GOSUB 7020:GOSUB 2020:GOSUB 7080
6460 RETURN
6470 x=x+36:y=311
6480 GOSUB 7020:GOSUB 2020:GOSUB 7080
6490 RETURN
6500 x=x+36:y=320
6510 GOSUB 7020:GOSUB 2020:GOSUB 7080
6520 RETURN
6530 '
7000 REM ******* Leger Lines *******
7010 '
7020 IF treble=0 THEN GOSUB 7160
7030 IF bass=0 THEN GOSUB 7200
7040 RETURN
7050 '
7060 REM ******** Draw Tails ********
     *
7070 '
7080 IF treble=0 THEN 7110 ELSE 7090
7090 IF y<222 THEN GOSUB 2100 ELSE GO
     SUB 2220
7100 RETURN
7110 IF y>346 THEN GOSUB 2220 ELSE GO
     SUB 2100
7120 RETURN
7130 '
7140 REM ***** Draw Leger Lines *****
7150 '
7160 y=y+63
7170 IF y<285 THEN MOVE x-9,288:DRAWR
      27,0
7180 IF y<267 THEN MOVE x-9,270:DRAWR
      27,0
7190 RETURN
7200 IF y=176 THEN MOVE x-11,180:DRAW
     R 27,0
7210 IF y>283 THEN MOVE x-11,288:DRAW
     R 27,0
7220 IF y>300 THEN MOVE x-11,306:DRAW
     R 27,0
7230 IF y>318 THEN MOVE x-11,324:DRAW
     R 27,0
7240 RETURN
7250 '
8000 REM ******* End Routine *******
8010 '
8020 CALL &BBFF:END
```

LIST 7

(Tape subscribers please note that MLIST67 contains the merged version of MLIST6 and 7).

```
  10 GOSUB 1030:GOSUB 5020
1030 BORDER 9:MODE 1
1040 b$="  qwertyuiop@[":c$="
       ":d$="2 45 789 -^":p=1
1050 INK 0,0:INK 1,26:INK 2,12:INK 3,
     3,6:PEN 1:PAPER 2:CLS
1060 '
5000 REM ******** Synthesiser *******
     **
5010 '
5020 ON BREAK GOSUB 6030
5030 GOSUB 5720
5040 WHILE NOT finished
5050 select$=INKEY$:IF select$="" THE
     N 5050
5060 note=INSTR(keystr$,select$)
5070 IF note>0 AND note<23 THEN note=
     note-4:GOSUB 5130:GOTO 5050
5080 IF note>22 THEN choice=note-22:G
     OSUB 5320
```

```
5090 WEND
5100 '
5110 REM ********* Play  Note *******
     **
5120 '
5130 frequency=440*(2^(octave+(note-1
     0)/12))
5140 harmonyfreq=440*(2^(octave+(note
     -14)/12))
5150 period=ROUND(125000/frequency)
5160 harmonyper=ROUND(125000/harmonyf
     req)
5170 SOUND 129,period,0,0,volenv,tone
     nv
5180 IF chorus THEN SOUND 132,period-
     1,0,0,volenv,tonenv ELSE SOUND 1
     32,0,0,0
5190 IF harmony THEN SOUND 130,harmon
     yper,0,0,volenv,tonenv ELSE SOUN
     D 130,0,0,0
5200 RETURN
5210 '
5220 REM ******** Change Octave *****
     **
5230 '
5240 IF cursor=1 THEN octave=octave+1
      ELSE octave=octave-1
5250 IF octave<-2 THEN octave=2
5260 IF octave>2 THEN octave=-2
5270 LOCATE 37,13:PEN 3:PRINT octave
5280 RETURN
5290 '
5300 REM * Alters Envelopes & Effects
      *
5310 '
5320 PEN 0:IF choice<6 THEN GOSUB 566
     0
5330 IF choice>6 AND choice<12 THEN G
     OSUB 5680
5340 IF choice=13 OR choice=14 THEN c
     ursor=choice-12
5350 ON choice GOTO 5360,5370,5380,53
     90,5400,5410,5420,5430,5440,5450
     ,5460,5470,5240,5240
5360 x=2:y=7:volenv=1:GOSUB 5510:RETU
     RN
5370 x=2:y=8:volenv=2:GOSUB 5510:RETU
     RN
5380 x=2:y=9:volenv=3:GOSUB 5510:RETU
     RN
5390 x=2:y=10:volenv=4:GOSUB 5510:RET
     URN
5400 x=2:y=11:volenv=5:GOSUB 5510:RET
     URN
5410 x=2:y=13:chorus=NOT chorus:GOSUB
      5550:RETURN
5420 x=20:y=7:tonenv=1:GOSUB 5610:RET
     URN
5430 x=20:y=8:tonenv=2:GOSUB 5610:RET
     URN
5440 x=20:y=9:tonenv=3:GOSUB 5610:RET
     URN
5450 x=20:y=10:tonenv=4:GOSUB 5610:RE
     TURN
5460 x=20:y=11:tonenv=0:GOSUB 5610:RE
     TURN
5470 x=20:y=13:harmony=NOT harmony:GO
     SUB 5580:RETURN
5480 '
5490 REM ****** Highlight Choice ****
     **
5500 '
5510 GOSUB 5660
5520 highlight$=UPPER$(select$)
5530 PEN 3:LOCATE x,y:PRINT highlight
     $
5540 RETURN
5550 PEN ((-3)*chorus)
5560 LOCATE 2,13:PRINT "<SPC>"
5570 RETURN
5580 PEN ((-3)*harmony)
5590 LOCATE 16,13:PRINT "<TAB>"
5600 RETURN
5610 GOSUB 5680
5620 highlight2$=UPPER$(select$)
5630 PEN 3:LOCATE x,y:PRINT highlight
     2$
5640 RETURN
5650 '
5660 PEN 0:FOR i=7 TO 11:LOCATE 2,i:a
     $=MID$(keystr$,i+16,1):PRINT UPP
     ER$(a$):NEXT
5670 RETURN
5680 PEN 0:FOR i=7 TO 11:LOCATE 20,i:
     a$=MID$(keystr$,i+22,1):PRINT UP
     PER$(a$):NEXT
5690 RETURN
5700 REM ********* Initialise *******
     **
5710 '
5720 SPEED KEY 255,255
5730 WINDOW 1,40,1,14:chorus=-1:harmo
     ny=-1
5740 WINDOW#1,1,40,24,25:PAPER#1,2:PE
     N#1,0
5750 ENV 1,=9,6000,2,0,100
5760 ENV 2,1,15,2,1,0,150,1,-15,2
5770 ENV 3,=8,310,5,0,30
5780 ENV 4,=14,650,10,0,12
5790 ENV 5,1,15,15,8,7,15,1,-7,15,1,5
     ,15,2,-2,15
5800 ENT -1,1,1,2,1,-2,2,1,1,2
5810 ENT -2,4,2,2,2,-4,2
5820 ENT -3,1,2,3,2,-2,3,1,2,3
5830 ENT 4,1,100,1,10,-10,1
5840 keystr$="1q2we4r5ty7u8i9op-@^[":+
     CHR$(16)+"zxcvb m,./\"+CHR$(9)+C
     HR$(&F0)+CHR$(&F1)
5850 octave=0:volenv=1:tonenv=0
```

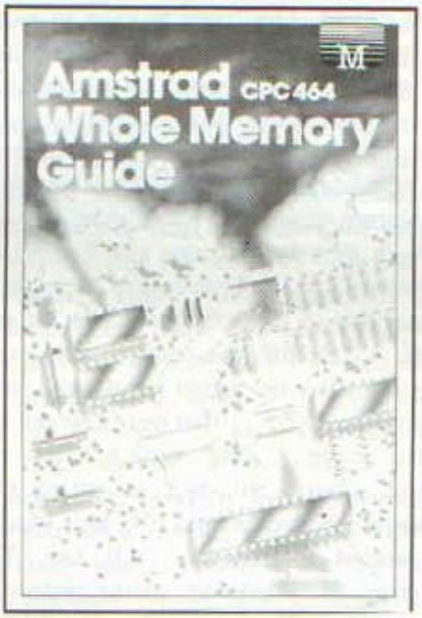
. . . .Continued on Page 32

Book Reviews

Whilst the Amstrad CPC464 Whole Memory Guide by Don Thomasson "provides the definitive guide for programmers", Making Music on the Amstrad CPC464 and 664 by Ian Waugh "allows users and their machines to write musical programs" - at least that is what the blurb says. Shane Kelly and Peter Campbell give us their opinions.

Amstrad CPC464 Whole Memory Guide

Reviewed by Shane Kelly



There are over 250 entry points to the CPC 464 firmware. If you had to find out what they all did by yourself you would spend the next 2 years trying to work it out. Don't bother. Author Don Thomasson has done it for you!

The reader is taken through the memory map of the 464 in the logical order of low memory restart functions through to the jumpblock and data areas pertaining to each section of the firmware. The low memory functions are explained in depth and some of the code is presented for your perusal. The explanation of how the 464 accesses ROM while running programs in RAM at the same address is excellent and should be understood by anyone who has a rudimentary knowledge of the Z80 and concepts of addressing.

After this comes an explanation of what happens when a full reset is issued, followed by the action of MC BOOT PROGRAM and MC START PROGRAM and then the rest of the machine pack follows in short order.

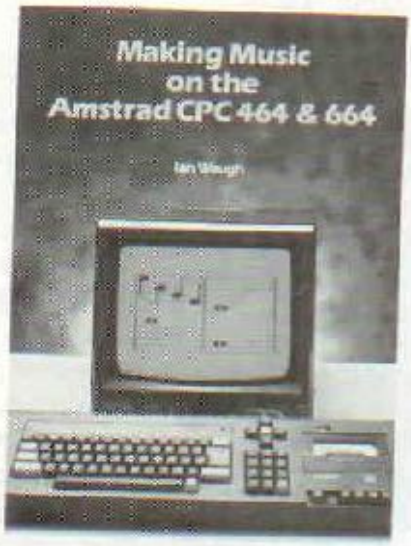
Then comes a clear and thorough look at one of the most powerful features of the 464 - the EVENT system. The interrupt path is explained in detail and how to set up your own 'events' becomes child's play. Next on the list is the DISPLAY SYSTEM which takes up 52 pages in it's entirety and is far too detailed to go into here. The SOUND MANAGER is given a short treatment and whilst not as comprehensive as the other parts of the book, is still quite acceptable given that the book is intended to show how the memory is set up and used in a standard way by the firmware. Following the sound section comes a short section on EXTERNAL ROMS. There are not too many external ROMS available in Australia but they are appearing in the U.K. and may work their way over here eventually.

Finally, we are given entry points to support routines for the BASIC interpreter maths routines. These are not easy to use according to Mr. Thomasson, but are presented anyway, along with a list of the BASIC keywords, their tokens and their addresses in the upper ROM. There is potential for experiment there!

In conclusion, a book for the interested (and knowledgable) BASIC programmer and a must for the serious 464 machine code 'hacker'.

Making Music on the Amstrad CPC464 and 664

Reviewed by Peter Campbell



Ian Waugh is a professional musician, who has written two other books on microcomputer musicmaking, "Making Music on the BBC Computer" and "Commodore 64 Music".

I was reminded of this and of the claims that some unscrupulous authors have turned their one book into a series by simply rehashing it on their word processors, when I read the caption on Figure 2.4: "The Keyboard, Note Names and SID 16-bit Numbers". The caption should, of course, have read PITCH numbers - SID is the sound chip in the Commodore.

This, however, seems to be his only indiscretion, for his book is packed with evidence that he has really studied the Amstrad's capabilities. He provides frequent cross-references to the CPC464's User Guide and even includes an appendix in which he advises: "I have tried to use meaningful variable names throughout the programs and all variables are in lower case. You may have noticed that the Amstrad automatically converts all keywords to upper case. This should help you to spot entry errors and assist in debugging."

Relevance to the Amstrad is only one of the qualities I look for in a book of this type. To be *useful* the book must contain *useful* information and be readable. Mr. Waugh's book scores handsomely on both counts. His knowledge of music, synthesisers and micro-computers is evident throughout and the detailed information is presented in an easily read style.

The book contains many listings. These illustrate points that the author is trying to convey and deal not only with the sounds that the computer can make but also with such things as wave forms. He presents a program that graphs envelope shapes and another that explores the computer's memory to show where the envelopes are stored. The latter is used to show how the computer can be tricked into accepting multi-section envelopes (i.e. having more than five sections).

Other listings include a synthesiser, which has four tone envelopes and four volume envelopes available at the touch of a key. It also can be shifted through a six-octave range. A toggle is provided to turn chorus effects on and off. A bass riff, presented as a separate listing, can be added, if desired, and suggestions for further development are also given. The listing for the synthesiser contained an error. To toggle chorus on and off, Mr. Waugh intended to use the TAB key. The listing as presented works with the space bar. The reason for this is that the listing should have contained a right pointing arrow (PRINT CHR$(1); CHR$(9)) which has 'got lost'.

Substituting "CHR$(9)" for the " " given in line 1240 will fix the problem.

A number of melodies are programmed and there is also a rhythm unit, with four rock rhythms, three 'fill' routines, a swing rhythm and a cha cha. Some sound effects and a program creating a soundscape of sea, surf and seagulls are also given.

Although Mr. Waugh says that he has refrained from using too many 'clever' programming techniques, the listings are, nevertheless, examples of goo programming and do contain som techniques of interest to more advance students of BASIC. For example would you have used:

```
chorus = NOT chorus
IF chorus THEN ....
```

to create a toggle? Up till the momen I saw it in the synthesiser program, the Boolean logic of it would have escaped me, but it is a simple and effective way of doing it.

In the closing chapters of his book, Mr. Waugh shows how to program the computer to compose music. This is not an easy task as he readily admits. "In the field of computer compositions we can safely assume that anything which sounds vaguely pleasant and does not make the listener squirm in his seat is a success. (The chapter on computer composition)..... goes a little further. I hope, and enables us to program the Amstrad to produce compositions in up to three voices which will be musically acceptable."

If you have some knowledge of the Amstrad computer and some of music and you want to go further into music programming then this is the book to get. Highly recommended.

Disc Cataloguer

by Petr Lukes

This program should make it easier to keep track of disc files. It establishes a catalogue for sixty discs on one side of the CAT disc. Since each side of a disc can contain a maximum of sixty-four files, there is no space for any files other than the program and its back-up and the sixty logs.

The discs are numbered 1 to 60 (or 61 to 120, 901 to 961, should it be necessary). The disc number is stored as the file name extension, and the file name is made up of the prefix CT and the date of the log, in the "yymmdd" format. The sides are identified as A or B, and each log contains the files of both sides of the particular disc. A separate file for each disc allows random access, eg. to read or write the log for disc number 10, we access the file "CT??????.010". Each log can be sent to the printer as it is created or subsequently read from the CAT disc.

All the logs on the CAT side can be searched serially for a particular file name. If the name is a null string, the search will list all the files in each log.

The program starts with logging-on of the CAT disc. It catalogues the disc into a specified buffer and stores the files which comform to the log format in the "ulog" array. This array is then updated each time a new log is written to the disc.

Creating a log is complicated by the possibility of files created under user numbers other than User 0. The disc is first catalogued to determine its format which is stored in the Disc Parameter Block. The four directory tracks are then read into the buffer, and the log is assembled into the "udisc" array. This is a fairly lengthy procedure, since files longer than 16k are spread over two or more directory entries, one entry for each completed 16k and one for any excess.

Each directory entry is 32 bytes long. Byte 0 has the user number or the value 229 if the file had been erased. Bytes 1 to 11 are the file name and extension, without the full-stop separator. Byte 15 contains the number of records covered by the directory entry. A record is 128 bytes long but files are written in blocks of 8 records, i.e. 1024 bytes. To get the total length of a file, whose catalogue may be spread over several directories, it is necessary to sum the number of records in each individual directory referring to the file. The total number of records is then converted into the number of blocks, rounded up. A block will not contain parts of two different files, so a file 1025 bytes long will occupy two blocks: its second block will have just the last byte and the EOF token (1a hex, 26 decimal), the remaining 1022 bytes will not be utilized.

Each file entry is assembled into the "udisc" array in the format of: byte 1=1-byte user number in hex (erased files are not logged), bytes 2 to 12=file name and extension, bytes 13 and 14=length of tile in blocks in hex. The array is then sorted in ascending order by an elementary bubble sort. Because the user number is the first character of each record and all records are the same length, the user number will automatically serve as the first key. This makes it easier to print the files under the different user numbers. The element zero of the array is used for a header record of the disc format, free space, and the number of files on each side.

The process is quite slow, since it involves extensive string manipulation and is likely to invoke one or more "garbage collections". But the delay is not excessive with up to about twenty files on the side, and most sides would probably not exceed this number. More time will be spent on accessing the

discs.

When a record is written to disc using the "PRINT#9,record" without the linefeed inhibitor, the operating system terminates the record with a NEWLINE and LINEFEED (codes 13 and 10 decimal). This would make each record 16 bytes long. If both sides of the logged disc contained the possible maximum of 64 files, the log would run to 3 blocks, and the disc capacity might be exceeded if a number of such logs existed. By inhibiting the linefeed and terminating each record with newline only, up to 128 bytes are saved and allow space for the header record for each side of the logged disc.

Reading and searching the log is quite elementary. The printout has two options: the file names can be printed across the page, or down the page in two columns. The latter option will use 66 lines with the maximum of 64 files; an 11 inch page will just accommodate this number of lines, but it would be safer to set the line spacing to seven lines per inch, if possible, to provide some top and bottom margins. In this format there is space for short hand-written comments next to each file name. There is no provision to log the files with their comments to the disc.

Two machine-language routines, stored in integer arrays, are used to access the directory tracks of each disc. The first one simply calls CAT with a buffer address. The second one, "rdir", is more complicated: it uses KL FIND COMMAND to find the address of the BIOS routine READ SECTOR (whose command name is CHR$(132), then reads the four directory sectors into the buffer for processing. Only the officially documented routines are used, so the program should be compatible with the 664 and 6128, although it was written on the 464.

DISC CATALOGUER

Examples of Printout showing Log both across and down

```
Disc no.1, logged 850930
 1a:S:free 136k:file(s)= 5
User 0
CATS     .BAK   7k   CATS     .BAS   7k   SECTOR   .BAK   6k   SECTOR   .BAS   6k
User 15
CATS     .BAS   7k
 1b:S:free 6k:file(s)=19
User0
ABASE    .BAK   16k  ABASE    .BAS   16k  ASHEET   .BAK   18k  ASHEET   .BAS 19k
ATXACM   .BAK   17k  ATXACM   .BAS   17k  CATALOGU .BAK   6k   CATALOGU .DSC  1k
CATALOGU .TXT   7k   DIR      .DSC   1k   DIR      .TXT   4k   DIRSEC   .BAK  4k
DIRSEC   .BAS   4k   DIRSEC   .TXT   4k   PAR      .      1k   PRAM     .BAK  5k
PRAM     .TXT   6k   SERIAL   .DSC   1k   SERIAL   .TXT   16k
```

```
Disc no. 1, logged 850930
1a:S:free 136k:file(s)=  5
0:  CATS     .BAK 7k
    CATS     .BAS 7k
    SECTOR   .BAK 6k
    SECTOR   .BAS 6k
15: CATS     .BAS 7k
```

```
1b:S:free 6k:file(s)= 19
 0: ABASE      .BAK   16k
    ABASE      .BAS   16k
    ASHEET     .BAK   18k
    ASHEET     .BAS   19k
    ATXACM     .BAK   17k
    ATXACM     .BAS   17k
    CATALOGU   .BAK   6k
    CATALOGU   .DSC   1k
    CATALOGU   .TXT   7k
    DIR        .DSC   1k
    DIR        .TXT   4k
    DIRSEC     .BAK   4k
    DIRSEC     .BAS   4k
    DIRSEC     .TXT   4k
    PAR        .      1k
    PRAM       .BAK   5k
    PRAM       .TXT   6k
    SERIAL     .DSC   1k
    SERIAL     .TXT   16k
```

```
100 PRINT CHR$(7):PRINT"DISC CATALOG
    UER 850928"
110 PRINT"Copyright 1985, P.Lukes, T
    OOWOOMBA, Q 4350, AUSTRALIA"
120 IF buff=0 THEN INK 0,24:INK 1,0:
    PEN 1:PAPER 0:MODE 2:buff=1:GOTO
     100
130 CLOSEIN:CLOSEOUT:IF buff>HIMEM T
    HEN 260
140 DEFINT f-t:DEFSTR u-z:first=1:la
    st=60:DIM ulog(last-first+1),udi
    sc(1,64),u(1),nr(64),rcat(4),rdi
    r(41):vf="DIS"'Disc format
150 PRINT"This disc must be in Drive
     A and the files on this side of
     the disc should be":PRINT"only
    `CATS.BAS` and `CTyymmdd.nnn`, w
    here nnn is between"first"and"la
    st:PRINT"Check the following cat
    alogue"
160 d=&BE42:db0=PEEK(d)+PEEK(d+1)*25
    6'Disc Parameter Block address f
    or drive A
170 d=&BE40:db1=PEEK(d)+PEEK(d+1)*25
    6'Disc Parameter Block address f
    or drive B
180 RESTORE 180:FOR L=0 TO 4:READ x,
    y:rcat(L)=VAL("&"+y+x):NEXT L:DA
    TA dd,5e,00,dd,56,01,c3,9b,bc,00
190 FOR L=0 TO 41:READ x,y:rdir(L)=V
    AL("&"+y+x):NEXT L:DATA dd,6e,00
    ,dd,66,01,e5,fd,e1,fd,6e,01,fd,6
    6,02,cd,d4,bc,d0,3e,c3,32,30,00,
    22,31,00,cd,0f,b9,c5,dd,5e,02,dd
    ,56,03
200 DATA 21,42,be,b7,ed,52,ed,52,4e,
    23,46,c5,fd,e1,fd,56,0d,fd,4e,0f
    ,dd,6e,04,dd,66,05,06,04,f7,30,0
    5,24,24,0c,10,f8,dd,6e,06,dd,66,
    07,77,c1,c3,18,b9
210 PRINT"Do not "CHR$(24)"I"CHR$(24
    )"gnore in case of read error":I
    NPUT"Press ENTER to log in CATS
    disc",z:OPENOUT"cats":buff=HIMEM
    :MEMORY buff-1:CLOSEOUT:CALL @rc
    at(0),buff
220 am=buff:WHILE PEEK(am)>0 OR PEEK
    (am+1)>0:z="":FOR b=am+1 TO am+1
    1:z=z+CHR$(PEEK(b)AND 127):NEXT
    b:L=VAL(MID$(z,9)):IF LEFT$(z,2)
    ="CT"AND L>=first AND L<=last TH
    EN ulog(L-first+1)=LEFT$(z,8)+"
    "+MID$(z,9)
230 am=am+14:WEND
240 INPUT"Is Drive B available (Y/n)
    ";z:dr%=-(z="Y"):IF dr%>0 THEN
    PRINT"Insert discs to be catalog
    ued into Drive B"ELSE PRINT"When
     prompted, remove CAT disc and i
    nsert disc to be catalogued, the
    n re-insert CAT disc"
250 WIDTH 255:ZONE 20:INPUT"Enter pr
    inter line width (default 80) ";
    z:IF z=""THEN pw=40 ELSE pw=VAL(
    LEFT$(z,3))\2:IF pw<20 OR pw>99
    THEN 250
260 PRINT"Logged discs:":FOR f=first
     TO last:IF ulog(f)>""THEN PRINT
     USING"###:";f;
270 NEXT f:PRINT:PRINT"1:Read log","
    2:Create log","3:Re-write log fr
    om memory","4:Search through all
     logs","5:End":INPUT z:g=VAL(LEF
    T$(z,1)):ON g GOTO 280,320,450,4
    80,660:GOTO 260
280 PRINT"READ":L=0:WHILE L<first OR
     L>last:PRINT"Enter number of di
    sc ("first"to"last")":INPUT y:L=
    VAL(LEFT$(y,3)):WEND
290 y=ulog(L-first+1):IF y=""THEN PR
    INT"No log for disc"L:GOTO 260
300 PRINT"Date of log:"MID$(y,3,6):O
    PENIN y:FOR s=0 TO 1:INPUT#9,z:P
    RINT:PRINT z:udisc(s,0)=z:FOR f=
    1 TO VAL(RIGHT$(z,3)):INPUT#9,z:
    udisc(s,f)=z:GOSUB 640[Rec]
310 NEXT f:PRINT:FOR g=f TO 64:udisc
    (s,g)="":NEXT g:NEXT s:CLOSEIN:G
    OSUB 540[Printer]:GOTO 260
320 PRINT"CREATE":WHILE LEN(date$)<>
    6:INPUT"Enter today's date in th
    e yymmdd format ";date$:WEND:L=0
    :WHILE L<first OR L>last:PRINT"E
    nter number of disc ("first"to"l
    ast")":INPUT z:L=VAL(LEFT$(z,3))
    :WEND
330 IF ulog(L-first+1)>""THEN PRINT"
    Log for disc"L"already exits, lo
    g date "MID$(ulog(L-first+1),3,6
    ):INPUT"Re-write (Y/n)";z:IF z<>
    "Y"THEN 260
340 IF dr%>0 THEN PRINT"Insert disc
    to be logged in Drive B":db=db1:
    :B ELSE PRINT"Remove CAT disc, i
    nsert disc to be logged":db=db0
350 FOR s=0 TO 1:PRINT"ENTER when re
    ady to log DISC"L CHR$(8)", side
     "CHR$(s+65):FOR f=0 TO 64:udisc
    (s,f)="":NEXT f:INPUT z
360 fi=0:o=0:POKE db+13,9:CALL @rcat
    (0),buff:q=PEEK(db+13):IF q>2 TH
    EN 430
```

```
370 re=9:zdir=CHR$(132):CALL @rdir(0
    ),@re,buff,dr%,@zdir:IF re>0 THE
    N 430
380 PRINT"Assembling log":am=buff:WH
    ILE (PEEK(am)<>&E5 OR PEEK(am+1)
    <>&E5)AND fi<64:IF PEEK(am)=&E5
    THEN 400 ELSE z=HEX$(PEEK(am),1)
    :FOR b=am+1 TO am+11:z=z+CHR$(PE
    EK(b)AND 127):NEXT b:fd=0:FOR f=
    1 TO fi:IF udisc(s,f)=z THEN fd=
    f:f=99
390 NEXT f:g=PEEK(am+15):IF fd<1 THE
    N fi=fi+1:udisc(s,fi)=z:nr(fi)=g
     ELSE nr(fd)=nr(fd)+g
400 am=am+32:WEND:o=0:FOR f=1 TO fi:
    g=(nr(f)+7)\8:o=o+g:udisc(s,f)=u
    disc(s,f)+HEX$(g,2):NEXT f
410 FOR f=1 TO fi-1:FOR g=f+1 TO fi:
    IF udisc(s,f)>udisc(s,g)THEN z=u
    disc(s,f):udisc(s,f)=udisc(s,g):
    udisc(s,g)=z
420 NEXT g:NEXT f
430 IF q>2 OR re>0 THEN v="Read Fail
    "ELSE v=MID$(vf,q+1,1)+":free"+S
    TR$(PEEK(db+5)-o-1)+"k:file(s)="
    +RIGHT$(" "+STR$(fi),3)
440 udisc(s,0)=v:PRINT udisc(s,0):NE
    XT s:y="CT"+date$+MID$(STR$(L/10
    00)+"0",3,4):IF dr%>0 THEN |A:db
    =db0
450 GOSUB 540[Printer]:PRINT CHR$(7)
    "Disc no."L:INPUT"Write log to d
    isc (Y/n) ";z:IF z<>"Y"THEN 260
    ELSE IF dr%<1 THEN PRINT"Remove
    logged disc, insert CAT disc"
460 INPUT"ENTER when ready to write
    log",z:PRINT"Writing "y:IF ulog(
    L-first+1)>""THEN |ERA,@ulog(L-f
    irst+1)
470 OPENOUT y:FOR s=0 TO 1:z=udisc(s
    ,0):PRINT#9,z:FOR f=1 TO VAL(RIG
    HT$(z,3)):PRINT#9,udisc(s,f)CHR$
    (13);:NEXT f,s:CLOSEOUT:ulog(L-f
    irst+1)=y:GOTO 260
480 PRINT"SEARCH":INPUT"Enter file n
    ame";z:z=UPPER$(z):x=z:g=INSTR(z
    ,"."):IF g>0 THEN x=LEFT$(z,g-1)
    +SPACE$(9-g):IF g<LEN(z) THEN x=
    x+MID$(z,g+1)
490 PRINT"Searching for " x:fd=0:FOR
     g=1 TO last-first+1:y=ulog(g):I
    F y=""THEN 520 ELSE OPENIN y
500 FOR s=0 TO 1:INPUT#9,z:FOR f=1 T
    O VAL(RIGHT$(z,3)):INPUT#9,z:IF
    INSTR(z,x)>0 THEN GOSUB 640[Rec]
    :PRINT"Disc "USING"###";g+first-
    1;:PRINT", side "CHR$(s+65)", lo
    gged "MID$(y,3,6)USING", user ##
    ";VAL("&"+LEFT$(z,1)):fd=-1
510 NEXT f:NEXT s:CLOSEIN
520 NEXT g:IF NOT fd THEN PRINT"Not
    found in log"
530 GOTO 260
540 'Printer
550 PRINT"Disc no."L:INPUT"Printer (
    Y/n) ";z:g=40:pr=0:IF z="Y"THEN
    pr=8:WIDTH pw*2
560 PRINT#pr,"Disc no."L CHR$(8)",lo
    gged "MID$(y,3,6):IF z="Y"AND pw
    >34 THEN INPUT"Down/Across ";z:I
    F LEFT$(LOWER$(z),1)="d"THEN g=p
    w:ZONE 1:GOTO 600
570 FOR s=0 TO 1:z=udisc(s,0):PRINT#
    pr,L;CHR$(8)CHR$(s+97)":"z:u="":
    FOR f=1 TO VAL(RIGHT$(z,3)):z=ud
    isc(s,f)
580 IF LEFT$(z,1)<>u THEN u=LEFT$(z,
    1):p=-(POS(#pr)>1):PRINT#pr,CHR$
    (13*p)CHR$(10*p)"User"VAL("&"+u)
590 GOSUB 640[Rec]:NEXT f:PRINT#pr:N
    EXT s:PRINT#pr:WIDTH 255:RETURN
600 PRINT#pr,L;CHR$(8)"a:"udisc(0,0)
    TAB(g)L;CHR$(8)"b:"udisc(1,0):u(
    0)="":u(1)="":FOR f=1 TO MAX(VAL
    (RIGHT$(udisc(0,0),3)),VAL(RIGHT
    $(udisc(1,0),3))):FOR s=0 TO 1:z
    =udisc(s,f):IF z=""THEN 630
610 p=s*g:IF LEFT$(z,1)<>u(s) THEN u
    (s)=LEFT$(z,1):PRINT#pr,TAB(p)US
    ING"##:";VAL("&"+u(s));ELSE PRIN
    T#pr,TAB(p)SPC(3);
620 GOSUB 640[Rec]
630 NEXT s:NEXT f:PRINT#pr:ZONE 20:W
    IDTH 255:RETURN
640 'Rec
650 x="  "+STR$(VAL("&"+MID$(z,13)))
    +"k":PRINT#pr,MID$(z,2,8)"."MID$
    (z,10,3)RIGHT$(x,5),:RETURN
660 CLOSEIN:CLOSEOUT:END
```

APOLOGY TO ENGLISH MAGAZINE SUBSCRIBERS

Due to circumstances beyond our control, the importation of both the January and February 1986 issues has been delayed, although the January issue is imminent. Apparently the delay has been caused by a change of printing company in the U.K.

Naturally, as soon as they are received, they will be despatched immediately.

Continued from Page 23

```
5860 PEN 1:PAPER 0:PRINT SPACE$(40)
5870 LOCATE 1,2:PRINT" A M S T R A D
       S Y N T H E S I S E R ";
5880 PRINT SPACE$(40):PEN 1:PAPER 2
5890 LOCATE 2,5:PRINT"Volume Envelope
     s":LOCATE 20,5:PRINT"Tone Envelo
     pes"
5900 PEN 0:PAPER 2
5910 LOCATE 2,7:PRINT"Z=ENV 1 (Piano)
     ":LOCATE 20,7:PRINT"M=ENT 1 (Vib
     rato 1)"
5920 LOCATE 2,8:PRINT"X=ENV 2 (Organ)
     ":LOCATE 20,8:PRINT",=ENT 2 (Vib
     rato 2)"
5930 LOCATE 2,9:PRINT"C=ENV 3 (Banjo)
     ":LOCATE 20,9:PRINT".=ENT 3 (Vib
     rato 3)"
5940 LOCATE 2,10:PRINT"V=ENV 4 (Tremo
     lo)":LOCATE 20,10:PRINT"/=ENT 4
     (Elecsynth)"
5950 LOCATE 2,11:PRINT"B=ENV 5 (Rever
     b)":LOCATE 20,11:PRINT"\=ENT 0 (
     Cancels)"
5960 LOCATE 2,13:PRINT"<SPC>=CHORUS":
     LOCATE 16,13:PRINT"<TAB>=HARMONY
     "
5970 PEN 3:LOCATE 2,13:PRINT"<SPC>":L
     OCATE 16,13:PRINT"<TAB>":LOCATE
     2,7:PRINT "Z":LOCATE 20,11:PRINT
      "\"
5980 PEN 0:LOCATE 31,13:PRINT"OCTAVE"
     ;:PEN 3:PRINT octave
5990 RETURN
6000 '
6010 REM ******** End  Routine ******
     **
6020 '
6030 CALL &BB00:CALL &BBFF:PAPER 0:PE
     N 1:END
```

Explanation of the codes appearing on subscriber labels (affixed to monthly deliveries of The Amstrad User.

Example: TAU/101/1234/M/1 JUN 86

- The Amstrad User CPC=English mag
- Internal sort code
- Subscription No.
- Magazine(M) or Tape(T) flag
- Renewal date

"I renewed my subscription to The Amstrad User but I haven't got the next magazine or a free copy of High Energy Programs for the Amstrad. What's going on?"

The above comment just about sums up some of the calls we have been receiving - but in all cases the fault was not ours. You see, the calls have come from people who returned their renewal cards late, that is, after we had produced the next month's mailing labels. (They are normally produced about five days before the month-end).

So, to avoid this happening to you, please try to return the completed renewal card as quickly as possible.

Thank you.

The 40cent Programs

by Ivor Joystick

Once upon a time, a sensible man bought an Amstrad computer.

Thirsting for information, he took out a subscription to The Amstrad User, the largest selling magazine for the Amstrad range in Australia.

He didn't need his machine to work out that, for less than $3* per month for 32 pages (and quite often more) of solid Amstrad information, he was getting programs for less than 40 cents each!

This made him live very happily ever after.

. . . but that's not the end of the story

Every month The Amstrad User is packed with a range of articles, programs, hardware and software reviews, utilities, User Group Information, Letters, Tutorials, Hints and Tips - all devoted entirely to the Amstrad computers.

And to save you the trouble of keying-in and correcting your typing errors, a cassette containing the programs appearing in the magazine each month is also available.

Make your Amstrad a success story by subscribing to The Amstrad User now.

* Based on yearly subscription of $35.

Please send me THE AMSTRAD USER for 12 months

Magazine only: $35 ☐ Magazine and cassette: $75 ☐ (PNG and NZ add $12 airmail)

Payment by: Cheque ☐ Bankcard or Mastercard ☐

Card number_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _Expiry date.....

Name ..... Phone.....

Address .....

..... Postcode .....

Signed ..... **Please start with Issue No.**.....

Return to THE AMSTRAD USER, Suite 1, 33-45 The Centreway, Blackburn Road, Mt. Waverley, Vic 3149 Tel 03-232 7055

(OVERSEAS PRICES ON APPLICATION TO ABOVE ADDRESS)