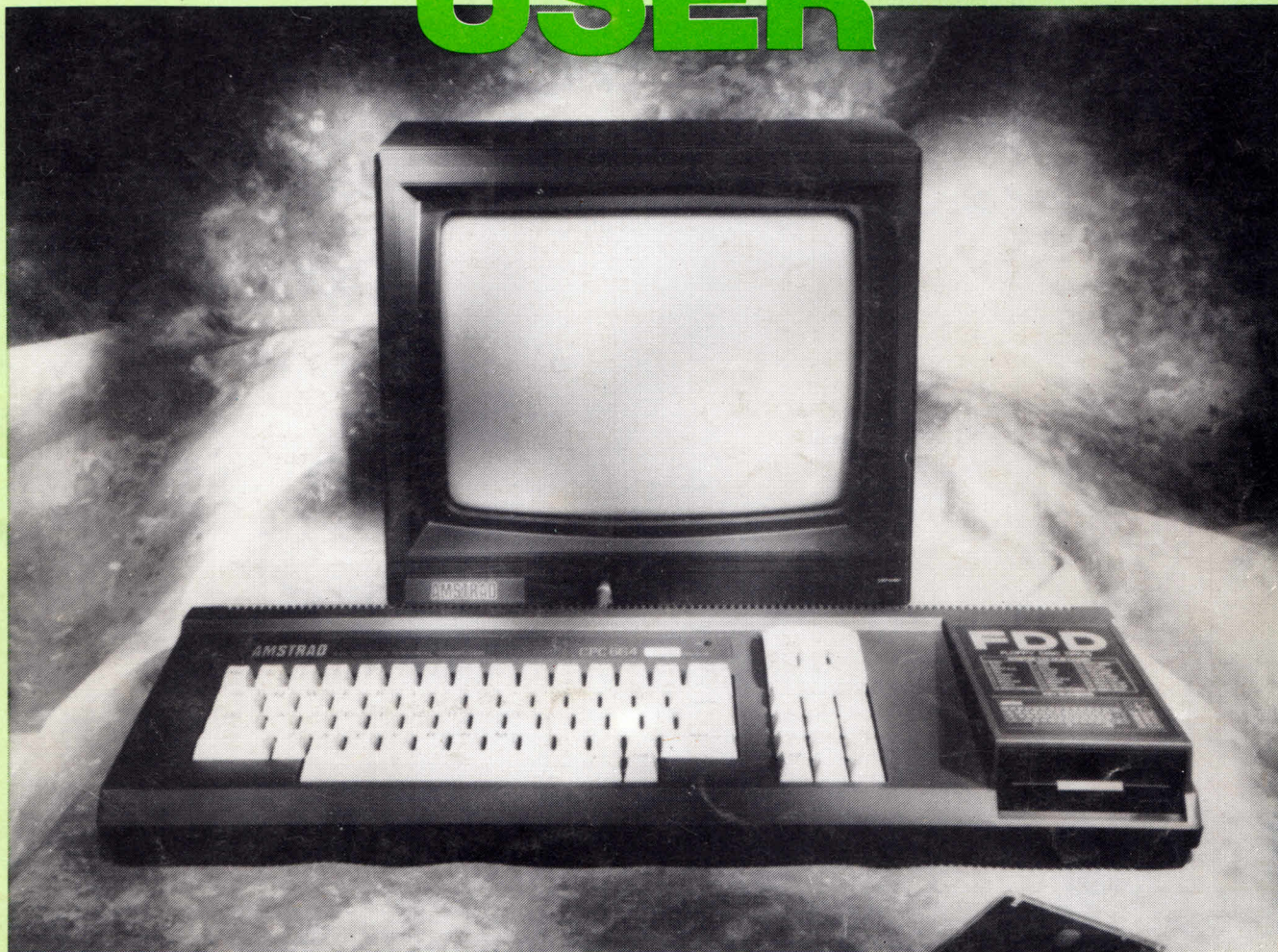


# THE AMSTRAD USER

Issue No. 6

\$3.00

July 1985



- REVIEW OF THE NEW CPC 664
- INTRODUCTION TO MACHINE CODE
- USER GROUP INFORMATION AND CONTACTS
- MACHINE CODE ROUTINES AND CP/M UTILITY

FOR THE NOVICE & EXPERIENCED USER

Registered by Australia Post – Publication No VBP7017

NEW

# Have you *really* discovered your Amstrad yet?

Want to get down to the nitty-gritty and start  
to see what you can *really* do with your  
Amstrad computer?

The following packages will get you started:

Teach yourself BASIC - Part 1	Soft 111
Teach yourself BASIC - Part 2	Soft 156
DEVPAC Assembler/Disassembler	Soft 116
Concise Basic Specification	Soft 157
Concise Firmware Guide	Soft 158
DDI-1 Firmware Guide	Soft 158a
PASCAL	Soft 155
Screen Designer	Soft 197
Pitmans Typing Tutor	Soft 924

Available from your Amstrad dealer

# THE AMSTRAD USER

Issue No. 6 July 1985

## CONTENTS

Editorial.....	2
Letters.....	3
BUZZLINES - a game.....	5
The Learning Centre - Introduction to Machine Code ....	8
The Trials of Tony Blakemore.....	9
Bytes and Pieces - Useful Machine Code routines.....	10
User Group Information and Contact List.....	13
Review of the CPC664.....	15
Discounted Books for Subscribers.....	18
The Ins and Outs of the Amstrad - a book review.....	22
FLASH - a game.....	23
Gencom - a CP/M utility.....	26
Retrieving Erased CP/M files.....	28
Typnig Errors - correcting some mistakes.....	29
History of Progammng Languages.....	30
Competition .....	32

---

All enquiries and contacts concerning this Publication should be made to The Amstrad User, Shop 2, 33 The Centreway, Blackburn Road, Mt. Waverley, Victoria 3149, Australia. [Telephone: (03) 232 7055].

The Amstrad User is published each month by Strategy Publications. Reprinting of articles in The Amstrad User is strictly forbidden without written permission. All rights reserved. Copyright 1985 by Strategy Publications.

The single copy price of \$3.00 is the recommended retail price only. The subscription rate (for Australia only) is \$30.00 for 12 issues of the magazine only, or \$70.00 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,

Some people may be aggrieved to discover that having just bought a CPC464, the 664 was announced. I'm not. As sure as eggs are eggs if I had bought a 664, an 864 or some other upgrade would have been released. Such is the development of micros - if you wait there will be something better. But if you always wait for something better, you'll never get anything! So I am content to use the amazing 464 until my skills require a more powerful tool, by which time I will replace it with a . . . . . 1264?

What I find encouraging is AMSTRAD's obvious commitment to producing a range of computers that are in the price range of most people's pockets, yet have the power and facilities of much more expensive machines. If the success of the 464 is repeated in future releases, they have nothing to worry about. For our sakes, I hope it will be.

This months' magazine sees a departure from the emphasis on BASIC that is normally presented. Machine Code takes the spot-light in this issue and will be featured more in the future. (Of course, I still rely on contributions from users and trust that you won't let me down). I can hear the cries of anguish from the 'new' programmers - "We've not got the hang of BASIC yet, now there is something else to learn!" But don't worry, it will be a gentle induction in an effort to add more armoury to your skills.

It is clear that a great number of readers of THE AMSTRAD USER are new to computers, hence the bias in their favour. Criticism, albeit infrequent, generally comes from the more experienced users looking for articles at a higher level. I would dearly love to publish such articles but, like the criticism, good contributions from this quarter are infrequent. I wonder why?

Finally, a reminder that there is now just six weeks left to get your competition entry to us, the closing date is 15th August. There will not be an extension. Don't forget to read Page 32 carefully before completing your entry.

See you next month,

Ed.

# Letters

I have just bought the book "Sixty Programs for the Amstrad CPC464" and having had a biorhythm program for my previous computer, I thought I would try the one from this book.

Unfortunately an error exists, and the program does not work without the following changes:

```
230 DAY=TOTAL2-TOTAL1
    (Delete the "1")
320 PLOT N-(P*C),167+167*s
    (N/(11.5*PI)),1
    (The "11.5" should be "23")
350 The "14" should be "28"
400 The "16.5" should be "33"
```

J. Steendam, Dee Why, NSW.

In the Amsoft "Master Chess" program there is a very annoying bug. In some situations the program allows a pawn on the 3rd rank to move 2 spaces; this is supposed to occur on the second rank only. How can I rectify this?

In your first publication of "THE AMSTRAD USER", you list a program which improves the pontoon program given in the instruction manual. It works perfectly for just one game but it doesn't keep a record of how many times you have beaten the computer. To do this I devised a very simple counting system to display the player's and computer's wins. In keeping with the same line numbers as given in that publication, here it is with line number changes only listed:

```
16 DIM suit$(4)
17 DIM pack(52)
70 (blank)
130 (blank)
500 LOCATE 12,17
```

```
510 PRINT"YOU'RE BUST":
    HOUSE=HOUSE+1
515 LOCATE 12,19:PRINT
    "WINS: PLAYER:";YOU;"
    HOUSE:";HOUSE
540 IF x$="y" THEN 20
670 LOCATE 12,17
680 PRINT "YOU WIN":YOU=
    YOU+1
700 LOCATE 12,17
720 PRINT"THE HOUSE WINS"
    :HOUSE=HOUSE+1
```

David R. Header, Rivett, ACT.

*We can only advise you to refer the problem to AWA who distribute this software in Australia as there is no way we can, nor would want to break into the software to trace the problem. On the other hand we could suggest you confine yourself to making legal chess moves, in which case the bug would not appear! Nevertheless, thanks for pointing out the problem - perhaps other owners of this game would care to comment.*

In Issue 4 (May 1985) of THE AMSTRAD USER one of your readers (JE, Brisbane) complained of a mistake in the listings for the game Pontoon from the book 'Sensational Games for the Amstrad'. Apart from the missing subroutine 11100, there are several other mistakes. The following patches will allow the game to run satisfactorily (although further development is needed to simulate all the betting rules of the casino game):

```
1. Add subroutine 11100:
11090 REM ** DEAL CARDS
    TO BANK **
```

```
11100 c=(RND(1)*51)=1
11110 IF av(c)=0 THEN 11100
11120 nc=nc+1
11130 bc(nc)=c
11140 av(c)=0
11150 RETURN
```

2. Add this line to give a more unpredictable deal:

```
14 RANDOMISE TIME
```

3. Add this line:

```
517 LOCATE 1,25:PRINT
    "Both players win":
    GOSUB 19500
```

4. Subroutine 15000 is not necessary and subroutine 15500 does not work as intended. Delete lines 14990 to 15510 inclusive.

5. In line 14515 replace GOSUB 15000 with GOSUB 18000.

6. In line 14520 replace GOSUB 15500 with GOSUB 16500.

7. Line 20010 replace with GOSUB 16500.

8. Line 18205 replace with RETURN.

9. Delete lines 17500 - 17510 which are not necessary and never called.

In addition I have amended line 14513 to replace 'END' with 'pl(q)=0:GOTO 14526' so that one player can quit without ending the game for the other player. I have also replaced the 'GOTO 10' in line 22005 with 'CLS:END' to give a natural game end when the Bank has won all the money.

I hope the above will prove useful to your readers.

P.H. Stehn, Garran, A.C.T.

Firstly let me congratulate you on producing and distributing a very informative and useful magazine. I am fortunate to have been a subscriber of 'The Amstrad User' since its first issue

and have found many of its hints and listings of great value.

Through the development of listings and ideas found in your magazine and elsewhere, I have evolved a very simple program that allows my young children to draw pictures on the monitor with the aid of a joystick. I intend further extending this program to allow more complicated pictures to be drawn.

My children have gained a lot of enjoyment from this activity but unfortunately as soon as the computer is switched off, their hard work disappears instantly.

As yet I don't have the knowledge needed to write a routine that allows the user to dump the screen graphics to a printer or to save it on to tape for later retrieval. I feel that this topic would be of interest to many budding programmers and I would appreciate it if any useful information on this topic could be forwarded to me and/or included in a future issue of your magazine.

Robert Wright, Bradbury, N.S.W.

*Here's a chance for you more experienced programmers to show us and Robert what you're made of by providing the solution.*

I would like to propose the idea of a marketplace in The Amstrad User. These are quite a regular feature of most English magazines, and while every care should be taken not to publish ads which would be in breach of copyright, such as the selling of software, I feel the market would soon open up for items such as printers, modems and other peripherals. Although things may start slowly, I'm sure people would soon realise the benefits of being able to sell or swap items that they find are not suitable for their own particular use, or things they may wish to upgrade, having learnt from experience. And since this is the Amstrad magazine, there is a pretty good chance many things would be compatible already.

Russell Cheek, Newcastle, NSW.

*We have had a number of requests to incorporate an exchange/market column in this magazine, but few requests with specific advertisements. To avoid a 'Catch 22' situation, we have included a space in this month's magazine for just such a column headed MICRO-MART.*

*We will let it run for a couple of months to gauge the response, so now it is upto you to take advantage of this opportunity to sell that old printer or the software you have been developing for the last six months.*

Firstly, I would like to hear from other Amstrad CPC464 users in the Geelong area who would like to establish a user group. I can be contacted by phone on (052) 50 2251 after 5 p.m. or by writing to me at:

10 Allambie St, Leopold, 3224

Secondly, could you answer the following questions?

1. Can you use the PRINT USING command to format strings in an orderly fashion, as it can be done with numbers?

2. Is it possible to use the SYMBOL command to form a character of more than one line (in a block)? I can use it OK in the form:

```
120 PRINT CHR$(128)+CHR$(129)
```

```
130 PRINT CHR$(130)+CHR$(131)
```

to create a character over two lines. Is there a simpler way of doing this?

3. Is it possible to use the ON BREAK routine with an "Input"? It works well during any other parts of my programs, but will not work while waiting for an input.

R.G. Butterfield, Leopold, Vic.

*To answer your questions in the same order:*

1. You can format strings with "!" or "&" or "\spaces". "!" will only print the first character of the string. "&" will print the whole string as is, and will ignore the internal zoning of the machine allowing you to print a string near the left edge of the screen and wrap around. Try these two examples:

```
10 LOCATE 37,10:PRINT "ABCDE
```

```
FGHI"
```

```
10 LOCATE 37,10:PRINT USING  
"&" "ABCDEFGHI"
```

*"\spaces\" will only print the characters selected from the start of the string. For example, if you try*

```
PRINT USING "\ \";"ABCDEFGH"
```

*and you place four spaces between the "\ " symbols, you will get ABCDEF. Note that the "\ " symbols are counted in the total of characters to be selected.*

2. You can only define an 8x8 character with the symbol command, but by re-defining characters available from the keyboard you could simply use: PRINT "AB← ←↓ CD"

*You may find that you will need to use the above PRINT USING "&" if the string is to be printed towards the right edge of the screen.*

3. The on-break gosub does not appear to work while waiting for an input as, at this point, the computer has temporarily returned to direct mode where only one press of the Escape key will break the program.

## MICRO-MART

*The Market Place for The  
Amstrad User Readers*

**THREE INCH DISCS** - now available on mail order in packs of five discs for \$42.50 plus \$2.50 airmail postage, from The Amstrad User. Bankcard or Mastercard accepted including phone orders on 03-232 7055.

**YOUR PROGRAMS** - if you think they are marketable could be sold through this column.

**MICRO-MART** is available for all readers to advertise for a nominal cost of \$10 per insertion. For further details ring Strategy Publications on 03-232 7055 or send your ad with cheque to Shop 2, 33 The Centreway, Mt. Waverley, Vic 3149.

534 8235

BOB BARTLETT

# Buzzlines

A Game from Roger Fraser

## About BUZZLINES

The core of this game came originally from a listing in the 'Advanced User Guide'. This enhanced version bears little resemblance to the original as many features have been added in an attempt to provide an appealing game of skill, but it does illustrate the fact that if you are prepared to spend a little time you can

turn a mediocore program into a more satisfying one.

The concept is simple. Using your joystick you must draw a YELLOW line between two RED lines. Any accidental touching of the red lines will result in a BUZZ that will delay you by about one second.

Once you press the FIRE button, the timer ticks away furiously, recording

the time you take to move from the left of the screen to the right. The elapsed time is measured in hundredths of a second. You may only move at right angles not diagonally - you don't want it too easy! If it gets too hard, or downright impossible to finish a screen, just press the Escape key to abort. You will then be returned to the start of a new and easier screen.

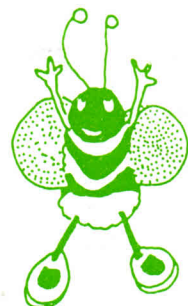
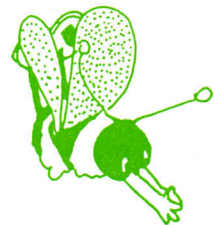
```
1 GOTO 8
2 SAVE "BUZZLINE.BAS" : STOP
3 REM SOFTWARE TYPE : Game - Action:FIRST ENTERED : 06/04/85
4 ' * LAST UPDATED : 27/04/85
5 ' COMMON NAME : Buzzlines
6 'DESCRIPTION : A computer version of the game where you have to move
  a ring along a bent wire without the ring and wire touching. In this
  version however, you
7 'have to draw a yellow line between two parallel red lines, without
  touching them. This program illustrates the TEST function.
8 CALL &BBFF : PEN 1 : PAPER 0 : : REM **** General Purpose Initialisa
  tion
9 'WARNING: Pressing the Escape Key does not break the program!!!!
10 MODE 0:BORDER 2,5
20 ON BREAK GOSUB 1410
30 PEN 3:PRINT #p:PRINT #p," B U Z Z L I N E S":PRINT #p:PRINT #p
40 PEN 2:PRINT #p,"In this game you":PRINT #p:PRINT #p,"have to draw a
  ":PRINT #p:PEN 1:PRINT #p,"yellow "":PEN 2:PRINT #p,"line between":PRI
  NT #p:PRINT #p,"two "":PEN 3:PRINT #p,"red"":PEN 2:PRINT #p," lines"
50 PRINT #p:PRINT #p,"from LEFT to RIGHT":PRINT #p:PRINT #p:PEN 15:PRI
  NT #p,"WITHOUT TOUCHING !!!":LOCATE 1,20:PEN 5:PRINT #p,"Press <ESC> I
  WICE if"":PRINT #p:PRINT #p,"screen is too hard.":PEN 4
60 FOR pause%=1 TO 18000:NEXT pause%
70 numpoints=4
80 GOSUB 340:REM **** Set pattern
90 first=0
100 GOSUB 440:REM **** Draw screen
110 oldelapsed=elapsed
120 t=TIME
130 WHILE px<630
140 GOSUB 780:REM **** Move
150 GOSUB 870:REM **** Display time
160 WEND
170 IF px>900 THEN 80
180 elapsed=(TIME-t)/300
190 GOSUB 1180:REM **** Scoring
```



```

200 first=1
210 GOSUB 920:REM **** 'Same Again?'
220 IF LOWER$(ans$)="y" OR ans$="X" THEN 100
230 ERASE y
240 GOSUB 1000:REM **** 'Harder Screen?'
250 IF LOWER$(ans$)="y" OR ans$="X" THEN numpoints=numpoints+2:GOTO 80
260 GOSUB 1060:REM **** 'Easier Screen?'
270 IF LOWER$(ans$)="y" OR ans$="X" THEN numpoints=numpoints-2:IF numpoints<1 THEN numpoints=1:GOTO 80:ELSE GOTO 80
280 GOSUB 1120:REM **** 'Giving up?'
290 IF LOWER$(ans$)="y" THEN GOTO 320
300 'no change in numpoints
310 GOTO 80
320 MODE 1:LOCATE 1,24:BORDER 1:PEN 1:PAPER 0:END
330 '
340 REM **** SCREEN CO-ORDINATES
350 era=1
360 DIM y(numpoints)
370 steppoint=INT(640/numpoints)
380 startpoint=steppoint-1
390 FOR points=0 TO numpoints
400 y(points)=RND*200+100
410 NEXT points
420 RETURN
430 '
440 REM **** DRAW SCREEN
450 SPEED INK 5,3:INK 11,6,13:INK 14,24:INK 15,10,23:PAPER 5:CLS:BORDE
R 3
460 WINDOW #1,1,20,1,3:PAPER #1,7:CLS #1
470 WINDOW #2,1,20,21,25:PAPER #2,7:CLS #2
480 MOVE 0,0:DRAWR 0,399,3:DRAWR 639,0:DRAWR 0,-399:DRAWR -639,0
490 MOVE 0,y(0)+20:points=0
500 FOR x=startpoint TO 639 STEP steppoint
510 DRAW x,y(points)+20
520 points=points+1
530 NEXT x
540 MOVE 0,y(0)+21:points=0
550 FOR x=startpoint TO 639 STEP steppoint
560 DRAW x,y(points)+21
570 points=points+1
580 NEXT x
590 MOVE 0,y(0)-20:points=0
600 FOR x=startpoint TO 639 STEP steppoint
610 DRAW x,y(points)-20
620 points=points+1
630 NEXT x
640 MOVE 0,y(0)-21:points=0
650 FOR x=startpoint TO 639 STEP steppoint
660 DRAW x,y(points)-21
670 points=points+1
680 NEXT x
690 px=10:py=y(0)
700 LOCATE #1,2,2:PEN #1,4:PRINT #1,"TIME ELAPSED ";:PEN #1,5:PRINT #1
,"00.00";
710 LOCATE #2,2,2:PEN #2,15:PRINT #2,"PRESS ";:PEN #2,11:PRINT #2,"FIR
E";:PEN #2,15:PRINT #2," BUTTON";:LOCATE #2,5,4:PRINT #2,"WHEN READY"
;
720 GOSUB 1330
730 IF ans$<>"X" AND ans$<>"y" THEN PRINT CHR$(7);:GOTO 720
740 LOCATE #2,2,2:PRINT #2,SPACE$(18)
750 LOCATE #2,2,4:PEN #2,2:PRINT #2," GO GO GO GO GO ";
760 SPEED INK 1,1
770 '
780 REM **** MOVEMENT ROUTINE
790 a=px:d=py
800 px=px-4*(JOY(0)=8)+4*(JOY(0)=4)
810 py=py-2*(JOY(0)=1)+2*(JOY(0)=2)
820 SOUND 2,(py/2-30),1,3
830 IF TEST(px,py)=3 THEN BORDER 3,6:INK 14,1,24:px=a:py=d:SOUND 1,100
0,20:FOR pause=0 TO 15:GOSUB 870:NEXT pause:BORDER 3:INK 14,24
840 PLOT px,py,14
850 RETURN
860 '

```

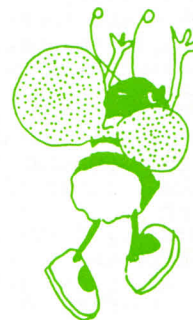
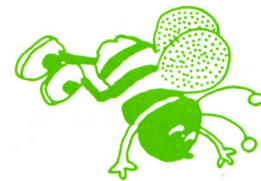
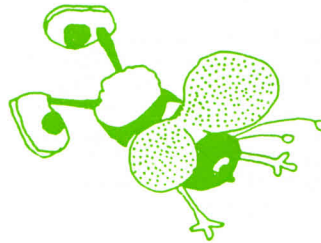




```

870 REM **** PRINT ELAPSED TIME
880 LOCATE #1,14,2:PEN #1,5
890 PRINT #1,USING "###.##";(TIME-t)/300;
900 RETURN
910 '
920 REM **** 'SAME AGAIN' ROUTINE
930 LOCATE #2,3,4:PEN #2,5
940 PRINT #2,"SAME AGAIN Y/N?";
950 GOSUB 1330
960 LOCATE #2,2,4
970 PRINT #2,SPACE$(18)
980 RETURN
990 '
1000 REM **** 'HARDER SCREEN' ROUTINE
1010 LOCATE #2,2,4:PEN #2,12
1020 PRINT #2,"HARDER SCREEN Y/N?";
1030 GOSUB 1330
1040 RETURN
1050 '
1060 REM **** 'EASIER SCREEN' ROUTINE
1070 LOCATE #2,2,4:PEN #2,13
1080 PRINT #2,"EASIER SCREEN Y/N?";
1090 GOSUB 1330
1100 RETURN
1110 '
1120 REM **** 'GIVING UP SCREEN' ROUTINE
1130 LOCATE #2,2,4:PEN #2,14
1140 PRINT #2,"YOU GIVING UP Y/N?";
1150 GOSUB 1330
1160 RETURN
1170 '
1180 REM **** SCORING
1190 LOCATE #2,2,2
1200 IF oldelapsed=0 THEN 1250
1210 IF first=0 THEN PEN 15:GOSUB 1280:GOTO 1260
1220 IF elapsed<oldelapsed THEN PRINT #2,USING "###.##";oldelapsed-elap
sed;:PRINT #2," SECS. BETTER";:GOTO 1260
1230 IF elapsed=oldelapsed THEN PEN #2,5:PRINT #2," NO IMPROVEMENT "
;:GOTO 1260
1240 IF elapsed>oldelapsed THEN PRINT #2,USING "###.##";elapsed-oldelap
sed;:PRINT #2," SECS. WORSE!";:GOTO 1260
1250 PEN #2,5:PRINT #2," FIRST TIME LUCKY ";:GOTO 1260
1260 LOCATE #2,2,4:PRINT #2,SPACE$(18)
1270 RETURN
1280 REM **** First time THIS screen, message
1290 IF elapsed<oldelapsed THEN IF elapsed+4<oldelapsed THEN PRINT #2,
"THAT WAS TOO EASY ";:ELSE PRINT #2,"SMARTIE AREN'T YOU ";
1300 IF elapsed>oldelapsed THEN IF elapsed-4>oldelapsed THEN PRINT #2,
"HARDER, WASN'T IT ";:ELSE PRINT #2,"YOU NEED PRACTISE ";
1310 RETURN
1320 '
1330 REM **** Y/N (INKEY$) ROUTINE
1340 ans$=INKEY$:IF ans$<>" " THEN 1340
1350 ans$=INKEY$:IF ans$=" " THEN 1350
1360 IF ans$="X" THEN 1390
1370 ans$=LOWER$(ans$)
1380 IF ans$<>"y" AND ans$<>"n" THEN PRINT CHR$(7);:GOTO 1340
1390 RETURN
1400 '
1410 REM **** ON BREAK ROUTINE
1420 numpoints=numpoints-2
1430 IF numpoints<1 THEN numpoints=1
1440 px=901
1450 LOCATE #2,2,2:PRINT #2,SPACE$(18);
1460 LOCATE #2,2,4:PEN #2,11
1470 PRINT #2," YOU GAVE UP !!! ";
1480 FOR pause%=1 TO 1700:NEXT pause%
1490 LOCATE 1,12:PEN 15
1500 PRINT CHR$(22)+CHR$(1)+" PRESS FIRE BUTTON"+CHR$(22)+CHR$(0)
1510 a$=INKEY$:IF a$<>" " THEN 1510
1520 IF era=1 THEN era=0:ERASE y
1530 RETURN

```



# The Learning Centre

## An Introduction to Machine Code

MACHINE CODE! Mumbo Jumbo that "real" programmers use? Some mysterious cult that only "advanced" computerists use? Not at all!

It is the purpose of these articles to gently introduce to the BASIC programmer the subject of machine code. I suppose that before you will become interested in this subject you must be convinced as to the value of machine code to you. By that I mean what can M/C do for you that BASIC can't? Let me count the ways:-

1) Speed - all (good?) arcade games are written in M/C because if they were written in BASIC the aliens would be flying through thick soup!

2) Graphics - Arnold's graphics are well implemented in BASIC, but are still limited by the BASIC interpreter.

3) Control - Arnold is unique in offering real time interrupt handling from BASIC, but so much more can be done with it if you have a knowledge of how to control the peripherals attached.

These three reasons are the ones usually quoted when extolling the virtues of M/C. On Arnold there is one reason that outweighs them all. When they designed Arnold they put "hooks" into the firmware (firmware is just the name for the programs already built into Arnold, i.e. the BASIC language) so that it becomes so easy to use M/C to create all those dazzling effects that make your eyes pop when you see them in other people's programs.

Now, before you run off and start key-bashing you are going to need a little reference material. First you will need a guide to those "hooks" into Arnold's firmware routines. There are two that I know of:- The CPC464 FIRMWARE MANUAL (scarce as

Hen's teeth) and a book called the "INS and OUTS of the Amstrad" which has the same information as the FIRMWARE GUIDE, but in a condensed format.

You will also need a reference work on the Z80 processor, mainly for its list of information (usually in an Appendix) called "opcodes". Opcodes is a contraction of two words, Operation and code and simply means the particular code that will cause the microprocessor to perform a certain operation. Listed alongside the opcodes are what is known as the MNEMONICS a funny looking word that means memory aid. These MNEMONICS are simply labels that make it easier to remember the codes and are used by a program called an ASSEMBLER that will "assemble" the opcodes from the mnemonics. A small example: Which is the clearer?

- a) 21 00 00 01 01 02 ES
- b) LD HL,0 LD BC,201, PUSH HL

While (a) is the opcodes represented by the mnemonic of (b) you will find it easier to remember (b) than (a).

OK, now, being armed with this reference material you are beginning to at least look like you know a M/C program from a BASIC keyword!

Let's now use our reference material to do some M/C programming. If you have the FIRMWARE GUIDE, turn to page 14.127 (PAGE 61 in the INS and OUTS). Look at the entry there. It is the specification of a routine called CAS START MOTOR. Place a program tape in Arnold's datacorder, press the play button. Now type the following:

```
CALL &BC6E <ENTER>
```

The cassette springs to life! Now type this:

```
CALL &BC71 <ENTER>
```

The cassette stops because we told it to!

Seriously though, this trivial example should serve to show you that because of the thoughtfulness of Arnold's designers, it is nearly that easy to control all of Arnold's M/C Routines.

Yes, I did say it was nearly that easy to control all of Arnold's M/C routines. There is an almost universal problem with interfacing M/C routines to BASIC, namely, the routines need data to work on and the BASIC program must make the data available. Arnold's designers have provided a method of passing data from BASIC to a M/C program, but it is tricky and best left until we have progressed a little further.

Instead, we will use a method that is slower but easier to understand, by placing the data to be passed in a location known to both BASIC and the M/C routine.

From reading your BASIC manual you will be aware of the BASIC commands - MEMORY and HIMEM. HIMEM, when used in the following manner, will give you a number that represents the highest location usable by BASIC.

```
xx = HIMEM
```

Now, to reserve some space for our data, we use the following statement:

```
MEMORY xx-10
```

This will reserve 10 memory locations that BASIC will be aware of but will not use. Our M/C routine can be made to get its data from this area. In addition, when the routine has finished, we can deposit any data that needs to be returned to BASIC.

To practice this concept of passing data from one routine to another, enter the following program:

```
10  xx=HIMEM:MEMORY=xx-10:
    BASE=xx-9
20  INPUT "Enter a number
    between 0 and 255",DATA
30  IF DATA <0 OR DATA >255
```

```
    THEN GOTO 20 ELSE POKE
    BASE,DATA
40  GOSUB 100
50  GOTO 20
100 PRINT "Press any key to see
    your number minus 5"
110 A$=INKEY$:IF A$="" THEN
    GOTO 110
120 PRINT PEEK(BASE)-5
130 RETURN
```

I will leave you to work out how the program functions - there is nothing difficult even for the novice BASIC programmer. However, note that the subroutine at line 100 is called, picks

up the data, works on it and returns. In the future, we will be substituting a M/C routine for the BASIC subroutine.

That about wraps up this month's article. Hopefully I have whetted your appetite for more. In the next article we will be delving a little deeper into Arnold to see just where he stores certain routines and how the Z80 processor works.

In the meantime, experiment with your reference books and Arnold and remember the other person's programs may "look" better than yours, but that's only because he knows how to utilise the full potential of the machine.

## The Trials of Tony Blakemore

*A column dedicated to the absolute beginner*

What a month it has been. User groups are springing up all over Australia and the combination of the knowledge from the members is producing some exciting discoveries. It is terrific to see the real beginners at last starting to join the groups and learning the basics of the CPC464 from experienced users and not just from books.

As last month's issue contained an excellent article on key re-defining, I will move on to a subject that really drove me crazy when I first started, namely Character re-defining. My first computer did not have the facility to change any of the characters so it came as quite a surprise to find out that on most computers you could.

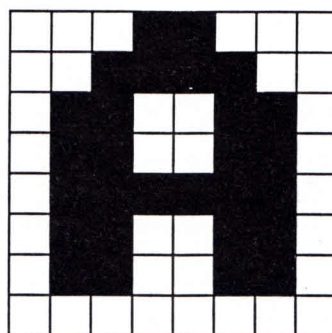
The first article I read was very interesting but all the numbers related to hexadecimal. (That's the type of number that computers and experts use just to confuse the beginner!). So off to another book to learn about hexadecimal numbers. At this stage I was convinced that the more I learnt the less I knew. Thank goodness the CPC464 understands ordinary numbers, so learning hexadecimal numbers at this stage is not necessary.

The CPC464 has in the ROM memory a program that sets up all of

the characters, required. This is done automatically when you switch the machine on. The list of characters and their ASCII codes, eg. chr\$(65) for the A, are all listed in the Users' Manual (Appendix III page 2).

Even though the complete set is quite comprehensive, there will be occasions when you will want to design your own characters. This is achieved by using SYMBOL AFTER commands (Chapter 8 page 47) to let the computer know from where you are going to change the characters. The SYMBOL command (Chapter 8 page 46) is then used to establish the new values.

128 64 32 16 8 4 2 1



The above grid illustrates the letter A

If you refer to the diagram above, you will see eight rows numbered one to eight. These are the positions in the SYMBOL command after the symbol number (the ASCII number) where you will place the new values to change the character.

Across the top of the grid are eight

more numbers. These numbers (added up come to 255) are used to establish the new values to be placed in the eight positions in the SYMBOL command.

To establish values for the letter A:

1. Look at the first row and you will see filled squares under the pixel numbers eight and sixteen (each square is a pixel). Now add the two values together and you get 24. This is the first value in the SYMBOL command after the character number (ASCII 65).

2. Count the second row;

$$4+8+16+32=60$$

This is the second value.

3. Count the third row;

$$2+4+32+64=102$$

This is the third value.

4. The fourth, fifth, sixth and seventh rows will result in the following values: 102, 126, 102, and 102.

5. Row eight has no filled pixels, so the eighth value is 0.

The SYMBOL command then for A is: SYMBOL 65,24,60,102,102,126,102,102,0

To change the letter B to an A, type the following small program:

```
10 SYMBOL AFTER 65
```

```
20 SYMBOL 66,24,60,102,102,126,
    102,102,0
```

Now press the B key and an A will appear. Practice changing various keys until you understand the principle thoroughly.

Next month we will have a look at producing graphic characters.

# Bytes and Pieces

## Useful Machine Code Routines from Sydney Brown

This month I will present you with some short but very handy machine code routines which you can add to your own programs.

I have included the source code for each routine which may help you to understand how each one works, as well as a BASIC loader which will allow you to experiment. The loader can be found at the end of this article and, as its name suggests, provides the BASIC code necessary to load each routine. Merely select the relevant lines of code for insertion into your own programs, or run the complete program to experiment. The machine code routines are held in the data statements in hexadecimal form

### Routine # 1

#### COPY SCREEN

```
21 00 C0 LD HL,#C000 Load source address (SCREEN)
11 00 40 LD DE,#4000 Load destination address(RAM)
01 00 40 LD BC,#4000 Load length to move (16k)
ED B0 LDIR Perform block move
C9 RET Return to Basic
```

This routine will copy the screen into a second 16k block of memory located at 4000H which can be useful to re-display a title page at game over or reset the game background after it has been corrupted.

### Routine # 2

#### RESET SCREEN

```
21 00 40 LD HL,#4000 Load source address (RAM)
11 00 C0 LD DE,#C000 Load destination address
(SCREEN)
01 00 40 LD BC,#4000 Load length to move (16k)
ED B0 LDIR Perform block move
C9 RET Return to Basic
```

This routine transfers all the data back to the screen. Both routines #1 and #2 could be modified to suit other requirements but TAKE CARE!!

### Routine # 3

#### GET SCREEN MODE

```
D5 PUSH DE Save address of chosen variable
CD 11 BC CALL #BC11 Call Firmware GET MODE
E1 POP HL Get variable address off stack
77 LD (HL),A Place mode value into variable
C9 RET Return to Basic
```

In some programs you may need to tell which mode the

computer is currently using while it is still running.

This routine will call the Firmware GET MODE routine and return the mode value into a chosen integer variable.

It can be used as follows:-

```
10 CALL SMODE,A% : REM A% will now contain the
current screen mode.
```

### Routine #4

#### FILL BLOCK

```
3E FC LD A,#FC Load encoded ink into A.
26 02 LD H,#02 Load Left column of block.
2E 03 LD L,#03 Load Top row of block.
16 0A LD D,#0A Load Right column of block.
IE 0B LD E,#0B Load Bottom row of block.
CD 44 BC CALL #BC44 Call SCR FILL BOX
C9 RET Return to Basic.
```

This routine will fill a rectangular area of the screen with a chosen solid or striped colour in an instant which can be useful when setting up backgrounds or borders around windows.

The encoding used by the Amstrad can be very confusing, but with a bit of time and some experimenting you should be able to get the colours you require.

The ink colour is encoded differently for each mode.

Mode 2 is quite simple as each pixel represents one bit of the byte.

Mode 1 is more complicated, as the byte is split into 4 sections.

Bits 3 and 7 determine the colour of the left most pixel in the byte, (Ink 0 to 3).

Bits 2 and 6 are next, then bits 1 and 5, and lastly Bits 0 and 4 control the colour of the right-most pixel.

Mode 0 is the most complicated as bits 1,5,3 and 7 control the left pixel and bits 0,4,2 and 6 control the right pixel.

Take particular notice of the order of bits as it does not seem to make sense.

The following is a diagrammatic representation of what I am trying to explain.

	Mode 0	Mode 1	Mode 2
Left Pixel	bits 1,5,3,7	bit 3,7	bit 7
			bit 6
		bit 2,6	bit 5
			bit 4
	bits 0,4,2,6	bit 1,5	bit 3
			bit 2
		bit 0,4	bit 1
Right Pixel			bit 0



CALL SSAVE Screen Save in one 16k Block.  
 CALL SLOAD Loads Screen in one 16k Block.  
 CALL FILL Fills Rectangular area with colour.

Poke BOT with Bottom character row.  
 Poke LFT with left column of area.  
 Poke RIT with Right column of area.  
 All poke values related to the fill routine are inclusive.  
 e.g. if you poke TOP with 2 and BOT with 7 then the block will be 6 characters high, 2 to 7 inclusive.

The variables TOP, BOT, LFT, RIT and COL are related to the fill routine.

Poke COL with encoded colour.  
 Poke TOP with top character row.

I am not quite sure what is causing the problem but if you try to put a 1 into TOP then you miss a bit of the top line of the block, it seems to be a fault somewhere in the Firmware routine.

```

10 MEMORY 32999:GOSUB 30:a%=0
20 STOP
30 REM
40 REM COPY SCREEN INTO MEMORY
50 REM
60 copy=33000:m=0
70 READ d:IF d<999 THEN POKE m+copy,d:m=m+1:GOTO 70
80 DATA &21,&00,&c0,&11,&00,&40,&01,&00,&40,&ed,&b0,&c9,999
90 REM
100 REM RESET SCREEN FROM MEMORY
110 REM
120 reset=33020:m=0
130 READ d:IF d<999 THEN POKE m+reset,d:m=m+1:GOTO 130
140 DATA &21,&00,&40,&11,&00,&c0,&01,&00,&40,&ed,&b0,&c9,999
150 REM
160 REM GET SCREEN MODE
170 REM
180 REM use with CALL SMODE,@A%
190 REM Screen mode value now in A%
200 REM
210 smode=33040:m=0
220 READ d:IF d<999 THEN POKE m+smode,d:m=m+1:GOTO 220
230 DATA &d5,&cd,&11,&bc,&e1,&77,&c9,999
240 REM
250 REM FILL RECTANGULAR AREA OF THE
260 REM SCREEN WITH COLOUR OR STRIPES
270 REM
280 REM poke COL with ENCODED COLOUR
290 REM poke TOP with TOP ROW of area
300 REM poke BOT with BOTTOM ROW
310 REM poke LFT with LEFT COLUMN
320 REM poke RIT with RIGHT COLUMN
330 REM
340 fill=33050:col=fill+1:lft=fill+3:top=fill+5:rit=fill+7:bot=fill+9:
m=0
350 READ d:IF d<999 THEN POKE m+fill,d:m=m+1:GOTO 350
360 DATA &3e,&f0,&26,&02,&2e,&03,&16,&0a,&1e,&0b,&cd,&44,&bc,&c9,999
370 REM
380 REM GET CHARACTER FROM SCREEN
390 REM
400 REM Locate cursor as per normal
410 REM use with CALL CHAR,@A%
420 REM Character value now in A%
430 REM
440 char=33070:m=0
450 READ d:IF d<999 THEN POKE m+char,c:m=m+1:GOTO 450
460 DATA &d5,&cd,&60,&bb,&e1,&77,&c9,999
470 REM
480 REM SAVE SCREEN IN ONE 16K BLOCK
490 REM
500 ssave=33080:m=0
510 READ d:IF d<999 THEN POKE m+ssave,d:m=m+1:GOTO 510
520 DATA &21,&00,&c0,&11,&00,&40,&3e,&16,&cd,&9e,&bc,&c9,999
530 REM
540 REM LOAD SCREEN IN ONE 16K BLOCK
550 REM
560 sload=33100:m=0
570 READ d:IF d<999 THEN POKE m+sload,d:m=m+1:GOTO 570
580 DATA &21,&00,&c0,&11,&00,&40,&3e,&16,&cd,&a1,&bc,&c9,999
590 RETURN

```

## User Group Information

This month it is our sad task to report the death of Norm Hart, the Treasurer of the Amstrad Southern Group. He died in a car accident on 11th June. As a founder member of the Southern Group, Norm was an avid Amstrad user both at group meetings and at home. He was dedicated to the group and carried out his official function in a quietly efficient manner. He was enthusiastic in teaching his children the use of the computer.

The funeral was held on 14th June at which a large number of relatives and friends attended.

Norm leaves a wife, son and daughter to whom we extend our sympathy.

### GROUP NEWS

Things appear to be a little quiet this month after the burst of information in the last issue which took nearly three pages. The User Group Contact list continues to expand with 44 people now listed as contact points. We look forward to Tasmania enlarging their section and would welcome some news from the Northern Territory.

#### South Australia (Grange)

The new title of this group is The Amstrad Computer Club Inc.(SA). They were officially incorporated in June. Chris Sowden is the new President - (08) 295 5923. Further details concerning their Constitution may be available next month.

#### AMSWEST, Perth

At the last meeting of the above group it was suggested that they contact some Eastern States User Groups with a view to exchanging news and programs on a personal basis.

Any group wishing to make contact with Amswest for further details should write to: Mrs. P.T. Ardron, 6 Weston Street, Carlisle, W.A. 6101.

#### Amstrad Eastern Users Group

Following the first meeting last month which went on for "just a little longer than planned", the attendance at the next meeting is expected to double to 50. Beginners, intermediates and experienced users are all catered for in separate groups at the meetings. The contact point is Tony Blakemore on 878 6212.

#### Western Amstrad Users Club

This is a new group, established to cover the west of Melbourne, which had its preliminary meeting recently. During these initial discussions, it was decided to meet twice each month at the Tottenham North Primary School, South Road, Braybrook from 6.30 p.m. The next two meetings are planned for 9th and 23rd July. For further details contact Mike McQueen on 312 5594

## User Group Contact List

### NSW

John Patterson	Lismore	(066) 21 3345
Chris Craven	Canowindra	(063) 44 1150
Paul Wilson	Moruya	(044) 74 3160
Frank Humphreys	Mummulgum	(066) 64 7290
R. Vijayenthiran	Newtown	(02) 519 4106
Hans Hill	Blacktown	(02) 671 2929
Martin Clift	Narrabri	(067) 92 3077
Mrs. D. Sparks	East Gosford	(043) 24 3342
Jim Owen	Uranga	(066) 55 6190
Chas Fletcher	Toongabbie	(02) 631 5037
Bruce Jones	Coffs Harbour	(066) 52 8334
Mark Kelloway	Barrack Point	(042) 95 1581

### ACT

Arthur McGuffin	Kambah	(062) 31 9437
Chris Rogers	Fraser	(062) 58 5749

### Vic

R.A. Russo	Richmond	(03) 428 4281
Tony Blakemore	Nunawading	(03) 878 6212
Martin Scragg	Pearcedale	(059) 78 6949
Don Leith	Brunswick	(03) 383 1498
Michael Prezens	Frankston	(03) 781 2158
Mrs. G. Chapman	South Clayton	(03) 551 4897
Mike McQueen	Braybrook	(03) 312 5594
Paul Walker	Heathmont	(03) 729 8657
David Carbone	Burwood	(03) 29 4135
Sue Kelly	Manangatang	(050) 35 1402
Alan Harris	Sale	(051) 44 1454
Ron Butterfield	Leopold	(052) 50 2251
Rod Anderson	Camperdown	(055) 93 2262

### QLD

Paul Witsen	Bulimba	(07) 371 9259
Mick O'Regan	Gladstone	(079) 79 2548
R.C. Watterton	Toowoomba	(076) 35 4305
D.F. Read	Ingham	(077) 77 8576
Kylie Telford	Goondiwindi	Calingunee246 (weekendsonly)
Michael Toussaint	Loganlea	(07) 200 5414
Tim Takken	Ipswich	(07) 202 4039
Steven Doyle	Caloundra	(071) 91 3147

### SA

Chris Sowden	Morphettville	(08) 295 5923
Lindsay Allen	Murray Bridge	(085) 32 2340
Rick Cable	Pt. Pirie	(086) 5967

### WA

Bob Harwood	Cooloongup	(095) 27 1777
Dave Andersen	6 Kitchener Rd Merredin, 6415	
Graeme Worth	Scarborough	(09) 341 5211
Tony Clitheroe	Morley	(09) 275 1257
Mrs. P. Ardron	Carlisle	(09) 361 8975
P.M. Nuyens	Waroona	(095) 33 1179

### TAS

Conal McClure	Scottsdale	(003) 52 2514
---------------	------------	---------------

# TWO SPECIAL ANNOUNCEMENTS

## The English CPC464 User

**ONE** - We have obtained a limited supply of back copies of the English CPC464 User. These are available on a first come first served basis at \$4.00 per copy (including postage) from Strategy Publications.

**TWO** - We can now offer a twelve month subscription starting from the July 1985 issue at a cost of \$45 (including postage) or backdate to January 1985 at a cost of \$40 (including postage) if we have sufficient stocks.

*Copies of the magazine will be flown in from the UK each month as soon as they are published and posted to you within 48 hours. However please allow at least 21 days for receipt of your first copy(ies).*

### HOW TO ORDER

Complete the relevant section of this order (or a copy) and return to Strategy Publications.

**Back Orders:** Please supply one copy of the English CPC464 User at \$4.00 per copy as indicated below:

JAN /FEB..... MARCH..... APRIL..... MAY..... JUNE..... Total: \$.....

*(Note: Jan/Feb issue is a single magazine)*

**12 Month Subscription:** Please register a subscription for the English CPC464 User as indicated below. I understand that I can only start my subscription with the Jan/Feb issue **or** the July issue.

JAN/FEB 1985 to JAN 1986 inclusive - \$40.00 including postage .....

JULY 1985 to JUNE 1985 inclusive - \$45.00 including postage .....

*(Subscriptions will automatically start at July 1985 if stocks of previous issues are exhausted)*

I enclose a cheque or debit my Bankcard or Mastercard for \$.....

Credit card No..... Expires .....

Name and Address (in capitals).....

..... Post Code.....

(Telephone orders accepted) Signed.....

### STRATEGY PUBLICATIONS

Shop 2, 33 The Centreway, Mt. Waverley, Victoria 3149

Telephone: (03) 232 7055



# A Review of the CPC664

Simon Anthony takes a look at 'Arnold' Mark II

There is no question that the CPC464 is an amazing machine within its range and it was fairly inevitable that an upgrade would be just a matter of time. In fact, installing a disc drive instead of a cassette was suggested at the 464 launch over a year ago! It was inevitable too that the 664 would be priced as competitively as its brother (now affectionately known as Arnold) - the green screen version is expected to retail at \$799 and the colour screen version at \$999.

The hardware consists of the main keyboard with computer and disc system, the choice of green or colour monitor, and the provision for a cassette recorder and a second disc drive to be attached.

The colour scheme has changed a little: the main keyboard cabinet and screen housing being in the same dark grey but the non-printing keys are now a light blue and the rest light grey. The numbers on the numeric keypad are preceded with the letter 'F' to denote function keys. All the keys are slightly dished.

The cursor keys are the most notable change apparently pandering to the trendy new design allowing larger keys for games enthusiasts to hit. (They still haven't put them in the right place for me - I'm left handed.)

Overall, the keys have a positive action with the minimum of touch, something to do with the membrane construction, but I haven't yet decided if I like the 'stepped height' arrangement of the rows of keys, probably because I have used the 464 for so long.

Naturally, the disc drive, which is located in the same place as the cassette unit on the 464, is taller, but smaller than a separate DDI-1 (disc drive



available for the 464). One very useful feature appears on the top of the disc housing, an inlay containing a list of the 27 colours available together with their reference numbers plus a diagram showing the various key numbers.

Having a disc drive instead of a 'datacorder' is like moving into the 'turbo charged' class. Anyone who has only tape facilities will know that to load a lengthy program normally gives you enough time to make a cup of coffee. With a disc drive, you can pay for it by the money you save on coffee! It takes just a second or so to load a program and cataloguing is almost instantaneous. So it was a joy to use. The discs are the same as those used in the 464, 3 inch compact floppies with each side capable of holding 169k of formatted space with CP/M system tracks attached.

You get a copy of Digital Research's CP/M operating system and Dr. Logo programming language free as part of

the 664 package. Also available is a modulator/power supply (an MP2) to link to a domestic TV if required. A standard cassette recorder is connected to a socket marked TAPE with a special cable (CL-1). The existing additional disc drive (FD1) can be connected to the 664 with a DI-2 cable.

Most current 464 software is, as they say, 'upwards compatible'. That means it should be able to run on a 664. However, do check first. Some software may use areas of memory that the new disc operating system now occupies, whilst others may break so many 'rules' as laid down in the Firmware Guide that they are unlikely to run on the 664 without a great deal of time and effort. Most reputable dealers and software houses should be able to advise you whether or not a particular piece of software is capable of being run on both machines.

While I am on the subject of software, I should mention that the 664

has a few new BASIC commands. Naturally, any programs which contain these cannot be run on a 464.

**MASK** - a graphics command which allows a template to be specified when drawing lines. This means that you can draw various dotted or dashed lines, even round corners if the template has been designed this way.

**GRAPHICS PEN** - allows you to set the ink for drawn graphics independently of the text ink, and specify an opaque or transparent background.

**GRAPHICS PAPER** - allows you to set the area behind the drawn graphics independently by allowing the selection of colour of the undrawn areas of dotted lines.

**FRAME** - this synchronises moving graphics with the display frame scanning frequency. (&BD19 on the 464) and provides much smoother movement of graphics.

**FILL** - from the graphics cursor position, allows you to fill an area of screen with a given colour to the drawn boundaries or the edge of the screen.

**COPYCHR\$** - lets you copy a character from a screen area into memory.

**CURSOR** - allows you to switch the cursor on or off.

**CLEAR PRINT** - allows you to clear the keyboard buffer.

**ON BREAK CONT** - a facility to make a program continue despite the fact that the Escape key is hit.

**DEC\$** - produces a number in decimal string form.

**DERR** - provides 'values' to disc errors.

There have been many enhancements to existing commands, graphics in particular. **MOVE**, **MOVER**, **PLOT**, **PLOTR**, **DRAW** and **DRAWR** can have a specified parameter to set the graphics ink mode. Using **XOR**, **AND**, **OR** or normal mode determines the way drawn graphics will interact with those already put on the screen.

You can draw a multicoloured line where each single pixel in a group of eight pixels is a different colour, using the **XOR** with a **MASK** command. You can also set the ink colour of the next point to be plotted or drawn with the enhanced **MOVE** and **MOVER** commands.

Documentation for the 664 is unquestionably better than the 464. For a start, the manual (or Users' Guide) is almost twice the size, part of which goes into much more detail in the areas of graphics and sound. The format, like the 464, guides the novice programmer

through a foundation course, then steps through program development, followed with a chapter documenting all the BASIC keywords with examples. The next three chapters take a look at the disc drive and offers an introduction to CP/M and AMSDOS (Amstrad Disc Operating System) and the Dr. Logo programming Language. Further chapters reveal information on control codes, error messages, key and joystick codes, memory map and pin-out/socket connection details amongst other things. The final chapters explain the 664 in more detail, general computing principles and provides a glossary and six games programs.

The CPC664 clearly does not replace the 464 but opens another market for one of the lowest priced disc based machines available. It would seem that AMSTRAD are broadening their horizons into the business area with this upgraded 464, coupled with the fact that they are looking closely at the possibility of producing a computer with an integrated modem.

There is no question of the 464 being dropped. Far from it, with the combination of the 464 and 664, AMSTRAD expect to sell 600,000 computers this year. Sounds like a lot of bleary eyes in the early hours!

*(Continued from Page 31)*

were prolific producers of languages. They produced Algol 60, Scalp (Self-Contained ALgol Processor), Dope (Dartmouth Over-simplified Programming Experiment) and finally BASIC (Beginners All-purpose Symbolic Instruction Code).

Joss (Johnniac Open-Shop System) was an early attempt to design an interactive language. It is no longer used.

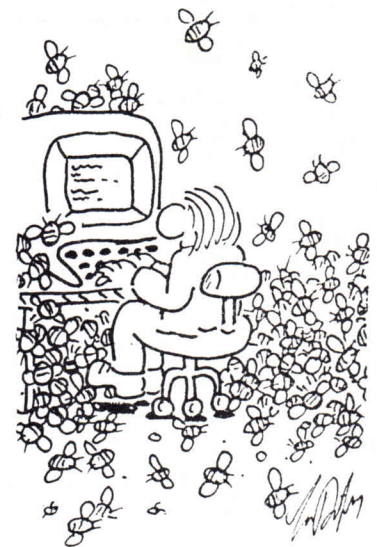
One of the newer languages, fast gaining popularity, is Pascal. It was developed by Niklaus Wirth in Switzerland and released in 1968. Several extensions have evolved and a number of compilers are available.

This has only skimmed the surface of the history of programming languages. Ladder was developed by

SRI International to deal with large complex data basis. Terry Winograd at MIT created Shrdlu in 1970 as a natural-language processor. Yale developed a language called Sam in 1977. Charles Moore created Forth to control telescopic equipment at Kitt Peak National Observatory. General Motors developed a language called Dyana, an extension to Fortran, to help measure vibrational and other dynamic systems.

The latest offering to emerge is Ada which was developed as the language to end all languages. Time will prove the durability of this latest offspring of a very fertile industry.

Next month I will attempt to compare and contrast BASIC, Fortran and Pascal.



ONLY A FEW MORE BUGS  
AND I'LL HAVE THIS  
PROGRAM WORKING RIGHT

# Subscribers Bonus Offer

## Valid to 31st July 1985

1. Choose and pay for any title from List 1 and receive any title from List 2 *absolutely free*.

OR

2. Choose and pay for any two titles from List 2 and receive any title from List 3 *absolutely free*.

### LIST ONE

Amsforth	\$49.95	Tasword	\$49.50
Graphic Magic	\$39.95		

### LIST TWO

3D Monster Chase	\$19.95	Alien Break-in	\$19.95
Amstrad Unlocked	\$19.95	Atom Smasher	\$19.95
Dragon's Gold	\$19.95	'Er*Bert	\$19.95
Global War III	\$19.95	Moors Challenge	\$19.95
Nightmare Park	\$19.95	Rollaball	\$19.95
Royal Quest	\$19.95	Soccer Boss	\$19.95
Tasprint	\$19.95	Trek	\$19.95

### LIST THREE

House of Horrors	Jigsaw Magic
Crazy Maze	Maths 1
Diamond Miner	Maths and Physics
Entrepreneur	Pathfinder
Extra Value Pack 1	Forgotten Planet
Pharaoh's Tomb	Shifty
Tee Off	

### ORDER FORM

Please supply from List 1 (title)..... at \$.....  
and include free of charge from List 2 (title).....

OR

Please supply from List 2 (titles)..... at \$.....  
..... at \$.....  
and include free of charge from List 3 (title).....

I enclose a cheque or debit my Bankcard or Mastercard for \$.....

Credit card No..... Expires .....

Name and Address (in capitals).....

.....Post Code.....

(Telephone orders accepted)                      Signed.....

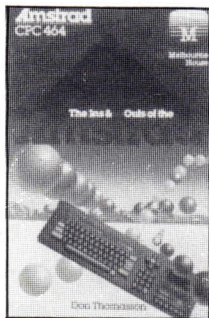
### STRATEGY SOFTWARE

Shop 2, 33 The Centreway, Mt. Waverley, Victoria 3149

Telephone: (03) 232 7055

# Discounted Books for subscribers of The Amstrad User

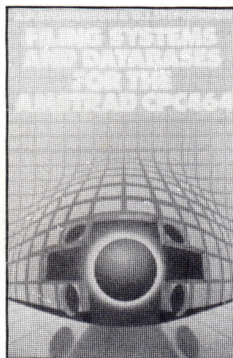
*The prices of titles below have been reduced by 10% for the benefit of subscribers. We are constantly reviewing the availability of books suitable for the CPC464, and only offer those which we consider worthwhile to users.*



## **The Ins and Outs of the Amstrad**

**\$17.95**

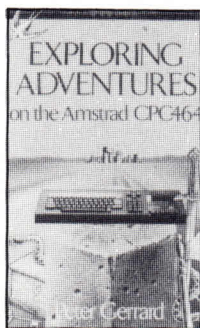
The next best book to the Firmware Guide. Deals clearly with how the CPC464 works with external equipment and studies the internal system. Although based on the cassette version of the CPC464, areas can be modified for disc systems. Sideways ROM, operating system, peripherals - it is all here.



## **Filing Systems and Databases for the Amstrad**

**\$22.45**

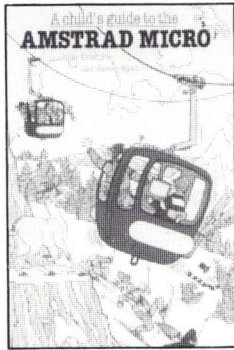
A useful book devoted to the storage, manipulation and retrieval of data using a cassette tape unit, much of which is relevant to disc drives. Contains major programs - a general purpose filing system and multiple choice tests - as well as a RAM-based serial filing system and searching and sorting routines.



## **Exploring Adventures on the Amstrad**

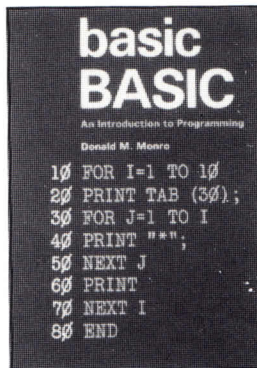
**\$17.10**

This book is of major interest to an adventure enthusiast who has always wanted to write one of his own adventures but thought it was too difficult. Carefully guides the programmer through room mapping, data structure, input routines and verb handling with a main game example called Underground Adventure.



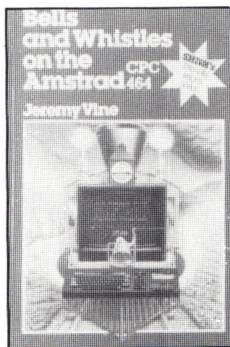
**A Child's guide to the Amstrad Micro** **\$ 8.95**

An excellent publication to introduce younger children to the power of the Amstrad. It is cleverly illustrated throughout to hold the readers attention. Five "characters" guide the new user and show not just what the computer can do, but what he/she can make it do.



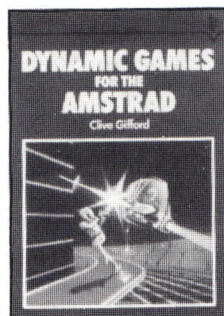
**Basic BASIC** **\$11.45**

A "mini-course" in BASIC which strives, in a serious manner, to teach new BASIC programmers the right way to structure their programs, and keep them falling into 'the hacker' mould. There is a sensible emphasis on mathematics and chapters on loops, graph plotting, tables and flow charts.



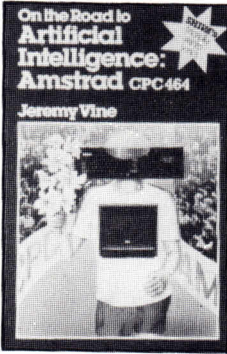
**Bells and Whistles on the Amstrad** **\$16.15**

This book contains nearly all you need to know about producing unusual, varied and exciting sounds on the CPC464. A rudimentary knowledge of BASIC is required. It will teach you to develop the sounds of different musical instruments, scales and ragtime, but above all, to produce your own sounds.



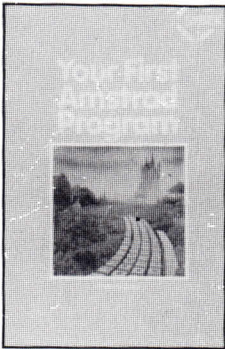
**Dynamic Games for the Amstrad** **\$17.05**

50 games covering a wide range from Arcade and Gambling themes to Graphics and Sound, with the emphasis on making the games interesting and fun. This book should give you plenty of ideas and routines to develop your own games if you wish, or keep you occupied by itself for many hours.



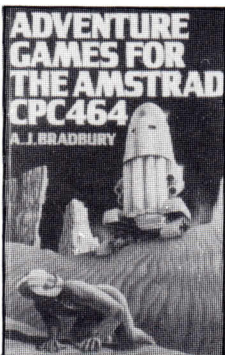
**On the Road to Artificial Intelligence** **\$17.95**

This teaches how to use the commands that are responsible for handling information and to combine them into programs that communicate with the user. It contains two programs to turn your computer into an intelligent companion, also an anti-crash course in BASIC.



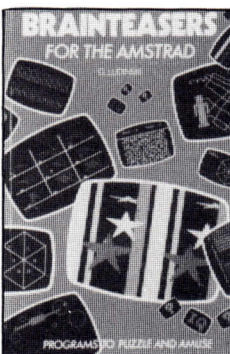
**Your first Amstrad Program** **\$18.85**

A publication somewhat different from the 'norm'. It is intended for the beginner who has no prior technical knowledge; is simple, straightforward and amusing. A pleasant introduction to BASIC for the young at heart.



**Adventure Games for the Amstrad** **\$17.95**

Whether you're an experienced programmer or a complete novice, you will find all the information needed to prepare, map and program complete adventures. It deals specifically with the range of LOCOMOTIVE BASIC instructions most likely to be needed when writing adventure programs.



**Brainteasers for the Amstrad** **\$17.95**

A collection of programs (28 in all) written to test your logic, general knowledge and mathematical skills. Many programs contain an IQ rating at the end. All exploit the graphics capabilities of the Amstrad.

### **The Amstrad Games Book**

A collection of exciting games for your Amstrad, including arcade-style space and combat games, maze games, impossible and dangerous missions for superheroes and more.

**\$14.35**

### **Amstrad Computing**

Takes you step by step, in detail, through the commands of your CPC464. You will find how business calculations are worked out, how vivid displays are obtained and how attention-catching sounds can be arranged.

**\$14.35**

### **Basic Programming on the Amstrad**

The first two chapters are for the complete beginner, the rest introduces the majority of the Basic commands available and deals with topics of more specialised interest, such as use of sound and how to create files and read data from them.

**\$17.95**

### **The Working Amstrad**

This book is a collection of solid application programs. The areas covered include home finance and tax, information storage and retrieval, household and diary management, creative graphics and effective display techniques.

**\$17.95**

### **40 Educational Games for the Amstrad**

The programs in this book have been designed to help the younger members of a family to handle the Amstrad and to increase their general knowledge - whilst enjoying themselves. Subjects include languages, mathematics and science.

**\$14.35**

### **Machine Code for Beginners on the Amstrad**

As the title suggests, this book is intended for the beginner wishing to learn how to use Machine Code on the Amstrad. Extensive use is made of the machine operating system allowing results from programs to be seen immediately.

**\$16.15**

### **The Amstrad CPC464 Advanced User Guide**

This book assumes you have got your CPC464 working and have already done some simple programming. It provides explanations on how the CPC464 communicates with external devices, on strings and characters, data structures, graphics and sound.

**\$15.65**

### **60 Programs for your Amstrad**

A massive software library for the price of a single cassette. Explosive games, dynamic graphics and invaluable utilities, this specially commissioned collection takes Basic to the limits and beyond.

**\$17.95**

### **An Amstrad CPC464 Compendium**

A fascinating selection of thirty games, each presented with a complete listing in Locomotive Basic, a helpful introductory description and a screenshot. There is something to appeal to the traditional and ultramodern computer gamesters alike.

**\$20.65**

**Please note:** The CP/M Handbook with MP/M, The CP/M User Guide and Logo Programming are currently unavailable. New stocks of these titles are expected in October 1985.

## **HOW TO ORDER**

*Send a list of the titles and quantities you require along with a cheque for the total **plus** \$5.00 postage and packing (regardless of the quantity you order) to:*

*Strategy Publications, Shop 2,  
33 The Centreway, Blackburn Road,  
Mount Waverley, Victoria, 3149*

*Bankcard or Mastercard orders accepted by phone on (03) 233 9227*

# The Ins and Outs of the Amstrad

A book review by Sonia Kelly

If you judge a computer's popularity by the number of books being written for it then Arnold is one of the most popular home computers on the market. A better way to judge is by the quality of the books published. Arnold wins using this method also. Having said that, I must say that this book will not be to everyone's taste. It is unashamedly a hardware buffs book that will allow those of you hooked on the fumes of hot solder to be high for weeks.

The book is divided into two sections - the INS and OUTS. Taking the INS first, the book takes the reader through the System Overview and memory systems fairly quickly and then delves into the more important parts of Arnold e.g. the I/O address map, the video gate array, the screen controller, the parallel peripheral interface and the printer port. In each of the sections listed above the reader is given information on the way Arnold is put together, how to utilise those functions that are usable (pointing out 'grey' areas that are either not fully defined or not enough is known about) and pinouts of the various ports where appropriate.

In each section clear diagrams are given, clearing away a lot of the fog generated by the use of 'jargon'. Also where appropriate, the reader is given those operating system calls that are directly applicable to the section being discussed.

Moving on, but still within the INS portion of the book, there is a discussion on the sound generator, keyboard and cassette. Each section is treated in a reasonable amount of depth and many of the ambiguities of the FIRMWARE GUIDE are dispelled, especially in the sound generator

department.

Now we come to the real meat of this book. The OPERATING SYSTEM. Pages 29 to 84 are given over to an excellent but not *too* detailed look at the firmware resident in Arnold. This section admittedly covers the same



ground as the FIRMWARE GUIDE but with a little less technical verbiage and hence the reader comes away with a much clearer understanding of the way the operating system actually knits all the pieces together. This section of the book is worth the price alone. However, that's not all you get!

The next section deals with the expansion capabilities of Arnold. A detailed look at the expansion port is given along with an explanation of the signal lines emanating from it. This is the 'OUTS' section. Here the hardware buff is given all the necessary information to interface his/her 'ultimate' peripherals to Arnold. Software support is also covered, but not to the same degree as the hardware side. Still, this book does not purport to be a software guide to interfacing. Also given is an explanation of external ROM's and details of an alternative printer port and an analogue port (using the resistive ladder technique).

All in all, a comprehensive coverage of much the same ground as the official FIRMWARE GUIDE from AMSOFT, but to my mind, in a clearer and simpler format.

A final word of caution before you rush out to purchase this book. It is not for beginners and you will need a good knowledge of machine code and some knowledge of hardware before this book can be of maximum value to you. There is some joy in it for the BASIC programmer and for those venturing to use the firmware routines already built into Arnold, but it is mainly aimed at the hardware 'hacker'. It is definitely easier to understand than the official firmware guide, and that being so, may be of more use to the BASIC programmer than the former heavy tome, but I cannot in all honesty recommend it solely on those grounds.

For the hardware buff, for the practiced BASIC programmer and for those whose curiosity extends to what actually makes an Arnold's personality, this book is recommended without hesitation.



# FLASH

A Game from Roger Fraser

## About FLASH

How good is your memory? In this 'Simple Simon Says' type of game, the computer FLASHes a random sequence of coloured squares into the four corners of the screen and then asks you to repeat the sequence using the keys:

R for Red    Y for Yellow

C for Cyan    B for Blue

The sequence is initially only one FLASH, but each time you correctly repeat the sequence, another flash is added. Each flash requires you to press the appropriate key (once per flash) to cause the sequence to be repeated exactly as you watched it. Beware of

multiple flashes of the same colour and wait before you repeat the sequence until you are prompted to do so.

You must complete the sequence as quickly as possible as the program will detect if you are too slow or just plain wrong. If you fail, the correct sequence will be repeated for you.

```
1 GOTO 9
2 SAVE "FLASH.BAS" : STOP
3 ' SOFTWARE TYPE : Game (reAction)
4 ' FIRST ENTERED : Jan '85
5 ' LAST UPDATED : 01/05/85
6 ' COMMON NAME : Repeat The FLASHing sequence
7 ' DESCRIPTION : The purpose of this program is to test your short-
term memory by repeating an ever increasing string of FLASHed squares
of colour.
8 ' SOURCE : Original idea from somewhere - Much Improved & Wor
king!!!
9 CALL &BBFF : PAPER 0 : PEN 1 : REM - General Purpose Initialisation
10 ' the actual program code begins here ...
20 DIM m$(7)
30 FOR x=1 TO 7:READ m$(x):NEXT
40 DATA "WATCH","REPEAT","Correct !","Wrong !"
50 DATA "The correct sequence" was ...,"Too s
low"
60 DATA "WAIT FOR IT ..."
70 MODE 0
80 nh$=STRING$(6,CHR$(143))
90 INK 0,0:INK 1,2:INK 2,18:INK 3,24:INK 4,6:INK 5,10:INK 6,4:INK 7,26
:INK 8,13,26:INK 9,12,25:INK 10,20
100 GOSUB 1380
110 REM **** New Game Begins Here
120 CLS:PAPER 0:BORDER 0
130 RANDOMIZE TIME
140 LOCATE 13,1:PEN 4:PRINT #p,CHR$(143)+CHR$(143)+CHR$(143)+CHR$(128)
;:PEN 3:PRINT #p,CHR$(143)+CHR$(143)+CHR$(143)
150 LOCATE 13,2:PEN 4:PRINT #p,CHR$(143)+"R"+CHR$(143)+CHR$(128);:PEN
3:PRINT #p,CHR$(143)+"Y"+CHR$(143)
160 LOCATE 13,3:PEN 4:PRINT #p,CHR$(143)+CHR$(143)+CHR$(143)+CHR$(128)
;:PEN 3:PRINT #p,CHR$(143)+CHR$(143)+CHR$(143)
170 LOCATE 13,5:PEN 10:PRINT #p,CHR$(143)+CHR$(143)+CHR$(143)+CHR$(128)
;:PEN 1:PRINT #p,CHR$(143)+CHR$(143)+CHR$(143)
180 LOCATE 13,6:PEN 10:PRINT #p,CHR$(143)+"C"+CHR$(143)+CHR$(128);:PEN
1:PRINT #p,CHR$(143)+"B"+CHR$(143)
190 LOCATE 13,7:PEN 10:PRINT #p,CHR$(143)+CHR$(143)+CHR$(143)+CHR$(128)
;:PEN 1:PRINT #p,CHR$(143)+CHR$(143)+CHR$(143)
200 PEN 7:LOCATE 2,10:PRINT #p,"PRESS:";
210 PEN 8:PRINT #p,CHR$(146)+CHR$(156)
220 SPEED INK 5,2
230 LOCATE 9,11:PRINT #p,CHR$(149)
240 LOCATE 9,12:PRINT #p,CHR$(149)
250 LOCATE 9,13:PRINT #p,CHR$(241)
260 PEN 11
```

```

270 LOCATE 9,15:PRINT #p,"1. SLOW"
280 LOCATE 9,17:PRINT #p,"2. FAST"
290 LOCATE 9,19:PRINT #p,"3. HELP"
300 LOCATE 9,21:PRINT #p,"4. END GAME"
310 PEN 9:LOCATE 3,2:PRINT #p,"BEST":LOCATE 4,4:PRINT #p,"OF":LOCATE 3
,6:PRINT #p,"LUCK"
320 z$=INKEY$:IF z$="" THEN 320
330 IF z$="1" THEN x%=40 ELSE IF z$="2" THEN x%=10 ELSE IF z$="3" THEN
GOTO 100 ELSE IF z$="4" THEN MODE 1:END ELSE PRINT CHR$(7);:GOTO 320
340 sc=0:ac=0:b$="":REM **** b$ holds sequence
350 WHILE ac=0
360 GOSUB 600:REM **** add letter to sequence
370 GOSUB 700:REM **** subroutine to display the sequence
380 GOSUB 850:REM **** try to repeat sequence
390 IF ti=x%*4 THEN CLS:BORDER 1,11:LOCATE 7,10:PEN 5:PRINT #p,m$(6):a
c=1:GOTO 460
400 IF aa$<>MID$(b$,zx,1) THEN CLS:BORDER 3,6:PEN 8:LOCATE 8,10:PRIN
t #p,m$(4):ac=1:GOTO 460
410 sc=sc+1
420 CLS:PEN 8:LOCATE 5,10:PRINT #p,m$(3)
430 PEN 1:LOCATE 5,13:PRINT #p,"Score is";sc
440 FOR pause%=0 TO 500:NEXT pause%
450 WEND
460 SOUND 1,4000,50,7
470 FOR pause%=0 TO 2000:NEXT pause%
480 CLS:BORDER 0:PEN 4:LOCATE 5,10:PRINT #p,m$(5)
490 FOR pause%=0 TO 2000:NEXT pause%
500 BORDER 0:CLS:PEN 8:LOCATE 8,10:PRINT #p,m$(1):GOSUB 700
510 FOR pause%=0 TO 1600:NEXT pause%
520 CLS:BORDER sc MOD 26
530 PEN 6:LOCATE 4,6:PRINT #p,"You SCORED :";:PEN 4:PRINT #p,sc
540 IF hsc>0 THEN PEN 6:LOCATE 6,9:PRINT #p,"H1 SCORE :";:PEN 4:PRINT
#p,hsc
550 IF hsc<sc THEN LOCATE 4,12:PEN 9:PRINT #p,"WELL DONE !!!":IF hsc=
0 THEN hsc=sc ELSE hsc=sc:PEN 9:LOCATE 16,9:PRINT #p,hsc
560 PEN 7:LOCATE 1,20:PRINT #p,"PRESS <";:PEN 8:PRINT #p,"SPACE";:PEN
7:PRINT #p,"> bar":LOCATE 7,23:PRINT #p,"to continue..."
570 aa$=INKEY$:IF aa$<>" " THEN 570
580 GOTO 110
590 END
600 REM **** subroutine to select a random letter and add it to b$
610 a=INT(RND(1)*4)+1
620 IF a=1 THEN a$="r"
630 IF a=2 THEN a$="y"
640 IF a=3 THEN a$="c"
650 IF a=4 THEN a$="b"
660 IF a=5 OR a=0 THEN GOTO 610
670 a$=LOWER$(a$)
680 LET b$=b$+a$
690 RETURN
700 REM **** display the sequence
710 IF ac<>0 THEN 750
720 FOR x=1 TO 1000 :NEXT x:REM **** time delay
730 dummy$=INKEY$:IF dummy$<>" " THEN 730 : REM **** clears inkey buffe
r
740 CLS:LOCATE 8,10:PEN 7:PRINT #p,m$(1)
750 FOR x=1 TO 1000 :NEXT x:REM **** time delay
760 CLS
770 LET xy=LEN(b$):xx=0
780 WHILE xx<xy
790 c$=MID$(b$,xx+1,1)
800 IF c$="r" THEN s=1 ELSE IF c$="y" THEN s=2 ELSE IF c$="c" THEN s=3
ELSE s=4
810 IF s=1 THEN GOSUB 1060 ELSE IF s=2 THEN GOSUB 1140 ELSE IF s=3 THE
N GOSUB 1220 ELSE GOSUB 1300
820 xx=xx+1
830 WEND
840 RETURN
850 REM **** Repeat sequence
860 CLS
870 dummy$=INKEY$
880 IF dummy$="" THEN 920
890 LOCATE 3,10:PEN 15:PRINT #p,m$(7)
900 FOR pause%=1 TO 1783:NEXT pause% : REM **** time delay of 1 second

```

```

910 dummy$=INKEY$:IF dummy$<>" " THEN 910 : REM **** clears inkey buffer
920 CLS:LOCATE 8,10:PEN 7:PRINT #p,m$(2)
930 FOR pause%=1 TO 891:NEXT pause% : REM **** time delay of 1/2 second
940 ti=0:CLS:zx=0
950 WHILE zx<>LEN(b$) AND ti<x%*4
960 ti=ti+1
970 aa$=INKEY$
980 aa$=LOWER$(aa$)
990 IF aa$<>"r" AND aa$<>"y" AND aa$<>"c" AND aa$<>"b" THEN 1030
1000 zx=zx+1
1010 IF aa$="r" THEN GOSUB 1060 ELSE IF aa$="y" THEN GOSUB 1140 ELSE IF aa$="c" THEN GOSUB 1220 ELSE IF aa$="b" THEN GOSUB 1300
1020 IF MID$(b$,zx,1)<>aa$ THEN 1040
1030 WEND
1040 RETURN
1050 REM **** draw graphic blocks
1060 REM **** top left
1070 PEN 4:CLS
1080 FOR x=1 TO 6
1090 LOCATE 2,x+2
1100 PRINT #p,hh$
1110 NEXT
1120 SOUND 1,478,40
1130 RETURN
1140 REM **** top right
1150 PEN 3:CLS
1160 FOR x=1 TO 6
1170 LOCATE 12,x+2
1180 PRINT #p,hh$
1190 NEXT
1200 SOUND 1,239,40
1210 RETURN
1220 REM **** bottom left
1230 PEN 10:CLS
1240 FOR x=1 TO 6
1250 LOCATE 2,x+12
1260 PRINT #p,hh$
1270 NEXT
1280 SOUND 1,119,40
1290 RETURN
1300 REM **** bottom right
1310 PEN 1:CLS
1320 FOR x=1 TO 6
1330 LOCATE 12,x+12
1340 PRINT #p,hh$
1350 NEXT
1360 SOUND 1,60,40
1370 RETURN
1380 REM **** instructions
1390 BORDER 0:MODE 1:LOCATE 15,1:INK 2,18,0:INK 3,15:PEN 2:PRINT "F L A S H"
1400 PEN 3:PRINT #p:PRINT #p:PRINT #p," This is a GAME OF MEMORY - SIMON SAYS"
1410 PRINT #p:PRINT #p," Watch the screen while coloured squares FLASH in a random sequence."
1420 PRINT #p:PRINT #p," You must try to duplicate the sequence, using the four blue keys (see below). These keys represent the four corners of the screen and colours that apply to them."
1430 LOCATE 10,17:PRINT #p,"Q W E R T Y U I O P"
1440 LOCATE 10,19:PRINT #p," A S D F G H J K L : "
1450 LOCATE 10,21:PRINT #p," Z X C V B N M , . / "
1460 PLOT 128,160,3:DRAWR 368,0:DRAWR 0,-112:DRAWR -368,0:DRAWR 0,112
1470 PRINT #p:PRINT #p:PRINT #p:PEN 1:PRINT #p," PRESS the <SPACE> bar to continue ..."
1480 LOCATE 16,17:PRINT #p,"R":LOCATE 20,17:PRINT #p,"Y"
1490 LOCATE 16,21:PRINT #p,"C":LOCATE 20,21:PRINT #p,"B"
1500 aa$=INKEY$:IF aa$<>" " THEN 1500
1510 aa$=INKEY$:IF aa$<>" " THEN 1510
1520 INK 2,18:INK 3,24
1530 MODE 0
1540 RETURN

```

# Gencom - a CP/M Utility

from Hans Hill

The utilities provided with the CP/M disc are now (dare I say it?) outdated, but as yet the excellent CP/M packages around are not available for the Amstrad. (Starcomm Systems are currently working on downloading 8" CP/M software onto the Amstrad 3" floppies, but for now let's use what we have).

GENCOM was developed to convert an object code file produced by the Hisoft DEVPAK (Soft 116) into an executable CP/M file. This allows the program to run by typing in the filename from CP/m without the need for BASIC loader programs or CALLs to a specific address. All Amstrad ROM routines can be used so nothing is sacrificed by using CP/M.

## HOW TO USE GENCOM

As you may know, AMSDOS creates a 128 byte header record that needs to be removed before CP/M can execute the file. What GENCOM does is to load the file generated by GENA 3 and resave it without the header record under (your filename).COM .

Now you have an executable CP/M file. To use the file so created:

A> (your filename) [Enter]

If your program is correctly written,

it will run.

To get GENCOM onto disc, run the Hisoft DEVPAK and start assembly at 100h - you can ignore the comments if you wish. Assemble the program using the A command. The object code is automatically saved to disc under the file GENCOM.ASM . After assembly, save the source file with:

Q 1,166,GENCOM.SRC

We now have to enter CP/M in the following manner:

A> DDT GENCOM.ASM - DDT will be loaded bringing GENCOM with it.

DDT VERS 2.2

NEXT PC

0300 0100 - you will see the DDT prompt.

-M180,2FF,100 [Enter]

-GO - this will return control to CP/M.

A> SAVE 2 GENCOM.COM

Now that we have GENCOM on disc let's demonstrate it by using Hisoft DEVPAK and entering the following routine:

ORG £100

LD B,26

LD A,"A"

LOOP: CALL £BB5A

INC A

DJNZ LOOP

JP £00

Assemble this little routine and save the object code to tape using

O,,TEST>OBJ

then exit to CP/M. After the prompt

A> GENCOM TEST.OBJ

GENCOM should burst into life and go to work. When complete you should see the 'PROCESS COMPLETE' message followed by the prompt.

Type in TEST [Enter]

and you should see 26 letters of the alphabet printed on the screen. GENCOM works !! Now it is up to you to write some really flash programs for other CP/M users. Get to it.

For those people who do not already have the Hisoft DEVPAK, Starcomm Systems can purchase, transfer onto disc and include GENCOM for a lousy \$75.00 with a 24 hour turn-round. If you already have DEVPAK but not on disc, send the front page of the manual for the transfer and GENCOM for \$20.00 .

Starcomm Systems, 48 Tara Road

Blacktown, NSW, 2148

(02) 672292

```
1 *H GENCOM developed for AMSTRAD users by STARCOMM SYSTEMS
2
3 ;The purpose of this little routine is to convert an assembly
4 ;program generated by GENA 3.1 into a (filename).COM file for
5 ;execution direct from CP/M.
6
7 ;File manipulation routines have been extensively documented
8 ;to allow users of this program to apply them in their own
9 ;programs.
10
11
12 ;GENCOM written by H.Hill of STARCOMM SYSTEMS...6th April 1985
13 ;intended for the public domain.
14
15 ;*****
16 ;
17 ;equates table
18 ;
```

```

005C      19 FCB: EQU £5C
0100      20 TPA: EQU £100
007C      21 FCBCR: EQU FCB+E20
1000      22 FXAREA: EQU £1000
0005      23 BDDS: EQU £05
0000      24 BOOT: EQU £00
000F      25 OPENF: EQU £0F
0014      26 READF: EQU £14
0010      27 CLOSEF: EQU £10
0009      28 PRINTF: EQU £09
0016      29 MAKEF: EQU £16
0015      30 WRITEF: EQU £15
0013      31 DELETF: EQU £13
          32 ;
          33 ;*****
0100      34 ORG TPA
          35 ;
          36 *T+ GENCOM.ASM
          37 ;
0100 010900 38 LD BC,£09 ;move filename up to storage area
0103 11AF01 39 LD DE,NEWFILE
0106 215C00 40 LD HL,FCB
0109 EDB0 41 LDIR
          42
010B 216800 43 LD HL,FCB+12 ;zero extents and records
010E 0617 44 ZEROF: LD B,23
0110 3600 45 LD (HL),0
0112 23 46 INC HL
0113 10F9 47 DJNZ ZEROF
          48
          49 ;OPENING THE DEFAULT FCB
          50 ;If your program takes a fileref as its first operand
          51 ;you can use the FCB set up at £5C in low storage.
          52 ;The CCP sets it up ready to open.
          53 ;
          54 ;*****CHECK THE BYTE AT £5D
          55 ;if it is a blank, then no operand was given.
          56 ;if it is a valid character, simply request service 15.
          57
0115 3A5D00 58 LD A,(FCB+1)
0118 FE20 59 CP £20
011A 113E02 60 LD DE,NOFILE
011D CAA701 61 JP Z,FINIS
          62 ;
          63 ;
0120 115C00 64 LD DE,FCB
0123 OE0F 65 LD C,OPENF
0125 CD0500 66 CALL BDOS
          67
012B 11F101 68 LD DE,OPENFL ;prints an error message if file not open
012B 3C 69 INC A
012C CAA701 70 JP Z,FINIS
          71
          72
012F 110010 73 LD DE,£1000 ;set up fix area
0132 D5 74 PUSH DE
          75
0133 115C00 76 RDSEQ: LD DE,FCB
0136 OE14 77 LD C,READF
0138 CD0500 78 CALL BDOS
013B FE00 79 CP 0
013D C25301 80 JP NZ,PROCES
          81
0140 3AD401 82 LD A,(RCCNT+1)
0143 3C 83 INC A
0144 32D401 84 LD (RCCNT+1),A
          85
          86 ;SETTING THE DATA BUFFER
          87 ;BDOS needs to know where to put the 128 byte data records
          88 ;once they have been read. In GENCOM the file is placed from
          89 ;address £1000 up, with the buffer address kept in DE.
          90 ;DE is changed to point to the next buffer area after each read
          91
          92
          93
          94
0147 D1 95 POP DE ;move records up to fix area for processing
0148 218000 96 LD HL,£80
014B 018000 97 LD BC,£80
          98
          99
          100
0151 18E0 101 JR RDSEQ
0153 102 PROCES:
0153 115C00 103 LD DE,FCB ;transfer new file name to FCB and zero all
0156 21AF01 104 LD HL,NEWFILE ;remaining bytes in FCB.
0159 012100 105 LD BC,33
015C EDB0 106 LDIR
          107
015E 115C00 108 LD DE,FCB ;delete any file of the same name
0161 OE13 109 LD C,DELETF
0163 CD0500 110 CALL BDOS
          111
0166 115C00 112 LD DE,FCB ;make a new file entry
0169 OE16 113 LD C,MAKEF
016B CD0500 114 CALL BDOS
          115 ;
016E 110202 116 LD DE,SPCFL ;if no space on disk print error message
0171 3C 117 INC A ;and abort.
0172 FE00 118 CP 0
0174 2831 119 JR Z,FINIS
          120 ;
0176 218010 121 LD HL,£1080 ;set up transfer back to DMA buffer area
0179 E5 122 PUSH HL
          123
017A E1 124 WRTSEQ: POP HL
017B 118000 125 LD DE,£80
017E 018000 126 LD BC,£80
0181 EDB0 127 LDIR
0183 E5 128 PUSH HL
          129
0184 115C00 130 LD DE,FCB
0187 OE15 131 LD C,WRITEF
0189 CD0500 132 CALL BDOS
          133
018C 111402 134 LD DE,WRTFL ;failed to write error message
018F FE00 135 CP 0
0191 2014 136 JR NZ,FINIS
          137
0193 3A7C00 138 LD A,(FCBCR)
0196 21D401 139 LD HL,RCCNT+1
0199 BE 140 CP (HL)
019A 20DE 141 JR NZ,WRTSEQ
          142
019C 115C00 143 LD DE,FCB
019F OE10 144 LD C,CLOSEF
01A1 CD0500 145 CALL BDOS
          146
01A4 112C02 147 LD DE,MSG ;prints successful message
01A7 OE09 148 FINIS: LD C,PRINTF
01A9 CD0500 149 CALL BDOS
01AC C30000 150 JP BOOT
          151
01AF 152 NEWFILE: DEFS 9
01BB 434F4D 153 DEFN "COM"
01BB 00000000 154 DEFB 0,0,0,0,0,0,0,0
01C3 00000000 155 DEFB 0,0,0,0,0,0,0,0
01CB 00000000 156 DEFB 0,0,0,0,0,0,0,0
01D3 0000 157 RCCNT: DEFB 0,0
01D5 20464149 158 READFL: DEFN " FAILED TO READ THE SECTOR $"
01F1 20464149 159 OPENFL: DEFN " FAILED TO OPEN $"
0202 204E4F20 160 SPCFL: DEFN " NO SPACE ON DISK$"
0214 20464149 161 WRTFL: DEFN " FAILED TO WRITE RECORD$"
022C 2050524F 162 MSG: DEFN " PROCESS COMPLETE$"
023E 204E4F20 163 NOFILE: DEFN " NO FILE REFERENCE GIVEN$"
0257 47454E43 164 DEFN "GENCOM VS1.0 by H.Hill"
026D 166 END
          Pass 2 errors: 00
          BDOS 0005 BOOT 0000 CLOSEF 0010
          DELETF 0013 FCB 005C FCBCR 007C
          FINIS 01A7 FXAREA 1000 MAKEF 0016
          MSG 022C NEWFILE 01AF NOFILE 023E
          OPENF 000F OPENFL 01F1 PRINTF 0009
          PROCES 0153 RCCNT 01D3 RDSEQ 0133
          READF 0014 READFL 01D5 SPCFL 0202
          TPA 0100 WRITEF 0015 WRTFL 0214
          WRTSEQ 017A ZEROF 010E

```

# Retrieving Erased CP/M files

A machine code utility for Basic programmers  
from Martin Scraggs and Tony Blakemore

In the March 1985 issue of the English CPC464 User was a very interesting article on CP/M Assembler. The article explained a little bit about CP/M COM files and adding a new file UNERA.COM - a file to retrieve CP/M files that had been accidentally erased. As an accomplished expert at accidentally erasing files, this had to be the best thing since sliced bread. Unfortunately, because of the nature of CP/M files which all start at hex100 (well below the normal area for the start of BASIC), a BASIC program was not available to produce this new file.

Talk about indian givers - here was the answer to my prayers and I could not get my hands on it. I contacted a good friend of mine Martin Scraggs who had helped out before when I had problems related to machine code. I wanted to find out if it was possible to enter the machine code using DDT on the CP/M disc. To complicate matters even further the listing contained Macros, a type of machine code subroutine, that made direct entry impossible. I left the problem with Martin who, amidst vague mutterings about "basic programmers who should not get involved with things they know nothing about", promised to see what he could do.

Listed below, thanks to Martin, is a method that will allow us BASIC programmers to enter and raise a new CP/M file called UNERASE.COM. Before entering the program which contains nearly two hundred entries I would stress the need for accuracy. If a mistake is made the whole procedure must be started again.

1. Place your CP/M disc into the disc drive and type |cpm and press ENTER. When A> appears type DDT and press ENTER. On the screen will appear:

```
DDT VERS 2.2
- (cursor here)
```

2. Type S100 and ENTER. The screen will now show:

```
DDT VERS 2.2
S100
0100 01 (cursor here)
```

3. This is the start address for entry of a new CP/M file. Listed below are the hex numbers that will enable the program to be entered. I would suggest taking a photocopy of this page and crossing off each number as it is entered.

Start at the left side of the listing and work to the end of the line. Type in each number and press ENTER. After three entries the screen will look like this:

```
DDT VERS 2.2
S100
0100 01 ED
0101 BC 7B
0102 0F 06
0103 C3 (cursor here)
```

You can ignore the second column, i.e. 01,BC,0F etc as they are only numbers that you are replacing. They will come up automatically as you press the ENTER key.

```
ED 7B 06 00 11 5C 00 0E 11 CD 93 01 3C C2 00 00
21 C3 01 22 9D 01 11 80 00 0E 1A CD 93 01 11 9F
01 0E 11 CD 93 01 3C 28 40 3D 87 87 87 87 87 16
00 5F 21 80 00 19 E5 23 06 0B 11 5D 00 1A 4E CB
B9 B9 23 13 20 18 10 F5 E1 E5 7E FE E5 20 0F 36
00 ED 5B 9D 01 01 20 00 ED B0 ED 53 9D 01 E1 11
9F 01 0E 12 CD 93 01 18 BD 11 C3 01 2A 9D 01 B7
ED 52 CA 00 00 D5 21 0C 00 19 7E 32 68 00 11 5C
00 0E 16 CD 93 01 D1 0E 10 CD 93 01 21 20 00 19
EB 18 D9 E5 D5 C5 CD 05 00 C1 D1 E1 C9 00 00 3F
3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 00 00 00 00 00
```

Now look at the entry opposite 01AB - if it is the first 00 after 3F then you have the right amount of entries. Press the CTRL key and C. This will return you to CP/M. When the A> appears type SAVE 1 UNERASE.COM & ENTER.

UNERASE.COM has now been placed on the CP/M disc. Type DIR to make sure that it is on the directory, if not the whole program will have to be done again.

To test the new file, filecopy any program onto the CP/M disc, make a note of its name and erase it. Return to CP/M and type UNERASE and the name of the file you have just erased. The program should now be restored to the directory. Run the restored program and check that it works. If it does, you have a very useful CP/M file.

UNERASE will only work if it resides on the same disc as the file accidentally erased. To use UNERASE on all your discs, filecopy it onto the disc when you format it. It only occupies 1K so you will not lose much space. As you become more confident and less accident prone you can leave it off the discs.

# Typnig Errors

## Correcting some mistakes so far detected

Errors are bound to occur from time to time, despite our efforts to check the contributions we receive before they go to press. We rely upon the readers of this magazine to be our back-stop to catch those we miss.

So we are grateful to those people who took the time to write and identify the mistakes.

3D ball (Page18, Issue 4) - Line 60 should read as follows  
Draw SizeX\*sin(A)+320,sizeY\*cos(A)\*sin(A\*0.95)+200

Down the Mine (Page28, Issue 3) - Line 4600 has a colon missing. It should be inserted following the quote at the end of the PRINT statement, ie ..hammer.":b14=1:GOTO 2450

MAP-CODE (Page10, Issue 5) - This problem will mainly concern disc drive owners. The program will lock if the drive is resident in the system. To correct this, merely change start=39000 in line 30 to start=HIMEM-5000.

Computerised Address Book (Issue 3) - The program listing has probably caused the most headaches. It was very difficult to distinguish between lower case L and number 1. Below is a listing of all lines containing lower case L's which we have changed to upper case. Anything else that looks like an 'el' should be read as a one.

```
1460 FOR L=1 TO size
1490 INPUT #9,namfld$(L):INPUT #9,strfld$(L):INPUT #9,twnfld$(L)
1500 INPUT #9,subfld$(L):INPUT #9,telfld$(L):INPUT #9,modfld$(L):INPUT #9,posfld$(L)
3110 FOR L=1 TO 1
3120 IF INKEY$<>" " THEN L=0
3130 NEXT L
3730 L=0
3750 FOR L=1 TO 1
3820 IF choi<1 THEN L=0
3830 IF choi>7 THEN L=0
3840 NEXT L
5160 FOR L=1 TO 1
5170 IF INKEY$<>" " THEN L=0
5180 NEXT L
6040 FOR L=0 TO size
6050 IF UPPER$(frt$)=UPPER$(twnfld$(L)) THEN town$(L)="yes" ELSE town$(L)="no"
6060 NEXT L
7015 IF namfld$(z)="" THEN L=size:GOTO 7030
7130 FOR L=0 TO size
7140 IF UPPER$(nrc$)=UPPER$(namfld$(L)) THEN c=L:L=size-1
7240 FOR L=1 TO 1
7260 IF a<1 OR a>6 THEN L=0
8020 FOR L=0 TO size
8030 IF UPPER$(dwr$)=UPPER$(namfld$(L)) THEN c=L:L=size
8060 FOR L=c TO size
8070 namfld$(L)=namfld$(L+1)
8080 strfld$(L)=strfld$(L+1)
8090 twnfld$(L)=twnfld$(L+1)
8100 subfld$(L)=subfld$(L+1)
8110 telfld$(L)=telfld$(L+1)
8130 modfld$(L)=modfld$(L+1)
8135 posfld$(L)=posfld$(L+1)
10255 IF n$="" THEN size=size-1:RETURN
10260 FOR L=1 TO LEN(n$)
10270 temp$=MID$(n$,L,1)
10320 NEXT L
10350 FOR L=1 TO LEN(n$)
10360 IF MID$(n$,L,1)=" " THEN s=L
10370 NEXT L
10400 FOR L=1 TO s-1
10410 IF ASC(MID$(n$,L,1))>64 THEN cnam$=cnam$+MID$(n$,L,1)
10420 NEXT L
10450 FOR L=s+1 TO LEN(n$)
10460 IF ASC(MID$(n$,L,1))>64 THEN snam$=snam$+MID$(n$,L,1)
10470 NEXT L
12050 FOR L=0 TO size
12060 PRINT #9,namfld$(L):PRINT #9,strfld$(L):PRINT #9,twnfld$(L)
12070 PRINT #9,subfld$(L):PRINT #9,telfld$(L):PRINT #9,modfld$(L):PRINT #9,posfld$(L)
12080 NEXT L
```

# History of Programming Languages

*Arthur Harris provides the background to the evolution of programming languages from 1940 to today. To new computer buffs it is historical, to the old it is nostalgic.*

In the early 1940's, computers like ENIAC were controlled by thousands of wires and switches. These physically controlled the flow of electrons through the computer and hence its performance. Each program required a different configuration and rewiring, from one program to another, took many hours. During the late 1940's, IBM developed the Card-Programmed Calculator. This was a large step forward. A set of pre-wired, special-purpose boards performed generalised functions. These boards made the CPC emulate a floating-point machine with built-in functions like square roots, sines and exponents. This arrangement was still not a saved-program computer. Each program had to be entered each time it was to be run.

These computers understood only machine code – a series of bit configurations that the computer converted to internal operations. Each code gave the machine one instruction, similar to throwing one switch or plugging in one wire on ENIAC. Developing and entering a substantial program took a long time and was very error prone, resulting in programs that were hard to debug.

As hardware became more sophisticated, it was realised that computers were useful and efficient tools. This realisation focused interest on automatic programming.

One effort to produce a system, or language, that would make it easier for the programmer to write programs that the computer would automatically convert to machine code through compilation, was centred around Massachusetts Institute of Technology. This effort produced the Whirlwind

computer which was developed between 1947 and 1951.

The first commercial venture into automatic programming was led by Dr. Grace Hooper of the Eckert-Mauchly Computer Corp. The result was Univac 1, which was programmed in mnemonic code known as Assembly Language. This was still rather clumsy and required several instructions to produce simple functions. Because Assembly Language is close to the language used by the computer, it is considered to be a low level language like machine code. Dr. Hooper's group continued along this line and laid the ground-work for most current high-level languages.

A high level language has a syntax far removed from the internal working of the computer. Ideally, it should be more easily understood by humans. Dr Hooper's efforts produced several compilers. The A2 compiler, the most widely used, uses a series of floating-point subroutines in main memory. This compiler depended on a sequence of compiling instructions, but acquired a pseudo-code after May 1954.

The Algebraic Translator, AT3 (called Math-Matic), was not commercially successful but a number of its concepts were used in the development of Algol. AT3 was not completed before Univac became obsolete as a scientific computer.

The B0 compiler (called Flow-Matic) contributed largely to the development of Cobol. Released in 1956, B0 relied on English-like syntax and was one of the first languages suitable for business applications.

At the same time, another Univac team was working on another compiler. Anatol Holt and William Turanski

produced the GP (Generalised Programming) system, based on hierarchies of library subroutines. This language was extended as GPX for the Univac II. This was the first computer language which primarily took account of the computer system, program segmentation and memory allocation.

Meanwhile, IBM released the IBM 701 one of the first commercial large-scale computers. The Speedcode language it used (developed in 1952) was easy to program but slow in operation. Next came the Pact system but, like AT3, it was not completed before the 701 became obsolete.

The development of a medium-sized magnetic drum encouraged software evolution. A number of interpretive languages appeared and the IBM 350 showed how proper data placement on the drum could optimise programming effort. Soap (Symbolic Optimiser and Assembly Program) utilised these features but was overshadowed by progress in other languages.

Dr. Al Perlis wrote the IT compiler at Carnegie Tech. IT took alphanumeric card input from the IBM 650 and produced a program in Soap. Later a program called Forttransit translated Fortran into IT, through Soap, to machine language.

Hardware technology continued to direct the progress of computer languages. The IBM 704 used magnetic core memory instead of electrostatic tube storage and incorporated many other improvements.

While developing the 704, in 1954, IBM placed John Backus in charge of a team to write a high-level, automatic program language for scientists,



mathematicians and engineers. The team included Irving Ziller, Harlan Herrick and Roy Nutt. Backus is quoted as saying the team "simply made up the language as they went along". In April, 1957, after some 25 man-years of work, the first Fortran (FORmula TRANslation) compiler was produced. At first it did not always work but after modification and improvement, it did produce executable code. This relatively easy-to-use language promoted practical use of computers.

Since then, several enhancements have produced new versions – Fortran II in 1957, and Fortran III and IV in 1962. IBM and a users group, SHARE, promoted use of the language. In May, 1962, a committee tried to develop standards for the language. In 1966, the American Standards Association (now the American National Standards Institute – ANSI) published its guidelines. The committee produced two standard Fortran languages – Basic Fortran for smaller computers and USA Standard Fortran for larger machines. Later, Fortran VI was developed and renamed PL/1.

Meanwhile, a European computer group, GAMM, was interested in developing an algebraic compiler for a variety of machines to try to produce a universal standard. Because of the international complications of the effort, Dr. Perlis and John Backus were named in its American faction. Although the group eventually discarded the idea of an international language, it published a report, The Preliminary Report on an International Algebraic Language, in the (northern) spring of 1958. The language, at first called IAL, became known as Algol. Later versions included Algol 58, Algol 60 and Algol 68.

Other algebraic languages, based on Algol, included Balgol (from Burroughs), Jovial (Jules Schwartz' Own Verion of the International Algebraic Language – from the Systems Development Corp.), Mad (from the University of Michigan) and Neliac (from the Naval Electronics

Lab.).

By mid-1959, most computers accepted Fortran and it seemed a universal language in USA. However, several of its features were awkward. Fortran suffered the drawback of being developed by IBM. This impeded its adoption as a universal language.

Further algebraic languages include Alpak and its successor, Alltran, from Bell Labs. SAC-1 was a large collection of Fortran subroutines. Algol was never successful in USA.

The US Government could afford complex systems and would require complex languages. It needed a new language that was compatible with many computers and was suitable for data processing, not scientific applications. In 1959, the Secretary of Defence called a meeting of representatives of manufacturers, users and academic institutions to form Codasyl (Committee on Data Systems Languages). Up till then, no one had paid much attention to business applications. Even B0 was limited by hardware inadequacy. Other attempts at business oriented languages include Aimaco (Air Material Command, 1959) and IBM's Comtran (1959).

Because of the urgency of the project, Codasyl selected two committees one to work on a short term solution based on existing languages and one to examine the recommendations of the first committee. The short term group recommended the development of a new language – Cobol. By the time the recommendation was presented, Honeywell had released Fact. A dispute ensued over these two languages, but eventually Cobol was accepted. The first report was released in April, 1960 and refinements resulted in Cobol 61. Fact was a powerful data processing compiler that relied on English syntax and worked for configurations as small as 4K. Cobol became widely accepted and became a commercial success. Cobol became the COmmon Business-Oriented Language. It was followed by Cobol 61 Extended in 1963, Cobol

65 and 68. ANSI approved a version in 1968 and approved revised standards in 1974, known as American National Standard Cobol 1974.

In 1963, 3 representatives of IBM and 3 members of SHARE formed the Advanced Language Development Committee. They aimed to produce a language called NPL (New Programming Language). This was changed to MPPL (Multi-Purpose Programming Language) and finally PL/1. This was released in April, 1964, revised in June, 1964 and refined until the release of 1976 PL/1 American National Standard.

IBM, meanwhile, was working on many other languages. One of them was RPG (Report Program Generator), released in the early 1960's and was followed by RPG II and RPG III.

The languages mentioned are only some of the 200 or more languages implemented since the 1950's with over 100 more no longer in use.

While scientific and data processing have dominated the computer world, scores of languages have been developed for list or string applications or specialised functions.

Amongst the oldest of the list-processing languages is IPLV (Information Processing Language V), released in 1958. It was one of the first to use memory cell lists linked with pointers. Other list processing languages include Sail, POP-2 and Slip, a descendant of FLPL, KLS, Threaded Lists and IPL-V. The most popular today seems to be Lisp.

Lisp 1.6, the latest version, has had a great influence on the development of Logo. Focus on string processing resulted in Comit, released in 1957. Snobol, a string oriented symbolic language was released in 1962.

The favourite language implemented on micro's is BASIC. It was a late development, since interpretive languages were not respected during the compiler period. Beginning in 1965, Prof. John Kemeny devised Darsimco (Dartmouth Simplified Code). Prof. Kemeny and Dartmouth

*(Continued on Page 16)*

# • COMPETITION •

*\$2500 worth of prizes to be won over four classes*

**Class 1**  
Best overall program  
Wins an AWA Video Recorder

**Class 2**  
Best amusement/adventure  
Wins a new DD1 disk drive

**Class 3**  
Best educational software  
Wins a new DD1 disk drive

**Class 4**  
Best business software  
Wins a new DD1 disk drive

## How to enter

Think about your program and map it out in a series of events or features. Write the program onto cassette based around these events and check that the program runs as intended. Once you are satisfied, send a copy of the cassette in a suitable envelope along with the following:

- 1 A brief summary of the program in 500 words or less.
- 2 A clear program listing if available.
- 3 A stamped, self addressed envelope of adequate dimensions if you would like your entry returned.
- 4 Your name and address.

You may make as many submissions as you want, but no entrant may win more than one prize.

## Conditions of Entry

- 1 All entries must run on a CPC464, and must include a cassette copy of the program (plus loading instructions where necessary), a brief summary of the program and its purpose and, if possible, a full listing.
- 2 All entries must arrive by 15th August 1985, and winners will be printed in the October edition of The Amstrad User.
- 3 The decision of the judges is final.
- 4 It is a condition of entry that all entrants have exclusive ownership of the copyright of the material submitted, and the winners agree to assign all copyright in the winning submissions to The Amstrad User. Where the entrant is more than one individual, then one person must be nominated and empowered to act on behalf

of the entire group. All entrants must undertake not to submit the same or a similar program to any other magazine, publisher or organisation until after the announcement of the winning entries.

- 5 We, The Amstrad User, may offer to publish programs other than the winners in the magazine or as commercial software, in which case we will agree terms on an individual basis with the author(s) concerned. We reserve the right to amend, alter or revise any program we publish.
- 6 No employees of The Amstrad User or Strategy Publications, or their relatives may enter this competition.
- 7 The Amstrad User cannot be held responsible for any loss or damage to any submission.
- 8 No entrant may win more than one prize.



They are disappearing *FAST* - and we wont be printing anymore! If you've missed any copies order NOW.

	MAG	TAPE		MAG	TAPE
Number 1 - Feb	.....	.....	Number 4 - May	.....	.....
Number 2 - Mar	.....	.....	Number 5 - June	.....	.....
Number 3 - Apr	.....	.....			

The cost of a single magazine is \$3.00 plus 50 cents postage and the cost of a single tape is \$4.50 plus 50 cents postage.

*Please send your order and remittance to:*  
The Amstrad User, Shop 2, 33 The Centreway,  
Mt. Waverley, Victoria 3149 Tel: (03) 232 7055  
(Bankcard and MasterCard accepted)

## RS-232 INTERFACES for CPC464

**Communicate with your Modem  
Talk to other computers  
Use Serial Printers  
Split Baud Rates  
Standard 25 way 'D' Connector**

**\$149.95**

Units are cased and include through connector for interstacking or connection of further add-ons (disc drive etc.)  
Literature supplied and Software on tape

Enquiries to:  
Strategy Computers  
Shop 2, 33 The Centreway, Mt. Waverley, Vic 3149  
Telephone: (03) 232 7055

# SAVE \$450

Q.

How many computer magazines would I have to buy to get at least 30 pages of information and program listings for the new Amstrad CPC464?

A.

On average, most popular magazines will devote 2½ pages to the AMSTRAD CPC464. This means you will need to buy 12 magazines at a cost of around \$40 per month, or **\$480 per year!**

Q.

Surely there must be a more sensible and cheaper way of getting the information I need?

A.

There is.  
THE AMSTRAD USER is a brand new monthly publication packed with articles, reviews, listings, hints etc. for Amstrad users only, and costs just **\$30 a year.**

Q.

How can I get a copy delivered to my home each month with an optional cassette containing all the program listings?

A.

By completing and returning this order.

Please send me THE AMSTRAD USER for 12 months

Magazine only: \$30  Magazine and cassette: \$70

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 4a, 33-45 The Centreway  
Blackburn Road, Mt. Waverley, Vic 3149 Tel 03-233 9227

(OVERSEAS PRICES ON APPLICATION TO ABOVE ADDRESS)